REPORT DOCUMENTATION PAGE

Form Approved

AFRL-SR-BL-TR-98-

0814

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REP(
	December, 1995	Final

4. TITLE AND SUBTITLE
USAF Summer Research Program - 1995 High School Apprenticeship Program Final Reports, Volume 14, Rome Laboratory

5. FUNDING NUMBERS

6. AUTHORS
Gary Moore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Research and Development Labs, Culver City, CA

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NI
4040 Fairfax Dr, Suite 500
Arlington, VA 22203-1613

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Contract Number: F49620-93-C-0063

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The United States Air Force High School Apprenticeship Program's (USAF- HSAP) purpose is to place outstanding high school students whose interests are in the areas of mathematics, engineering, and science to work in a laboratory environment. The students selected to participate in the program work in an Air Force Laboratory for a duration of 8 weeks during their summer vacation.

14. SUBJECT TERMS
AIR FORCE HIGH SCHOOL APPRENTICESHIP PROGRAM, APPRENTICEDHIP, AIR FORCE RESEARCH, AIR FORCE, ENGINEERING, LABORATORIES, REPORTS, SCHOOL, STUDENT, SUMMER, UNIVERSITIES

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

UNITED STATES AIR FORCE

SUMMER RESEARCH PROGRAM -- 1995

HIGH SCHOOL APPRENTICESHIP PROGRAM FINAL REPORTS

VOLUME 14

ROME LABORATORY

RESEARCH & DEVELOPMENT LABORATORIES

5800 Uplander Way

Culver City, CA 90230-6608

Program Director, RDL
Gary Moore

Program Manager, AFOSR
Major David Hart

Program Manager, RDL
Scott Licoscos

Program Administrator, RDL
Gwendolyn Smith

Program Administrator, RDL
Johnetta Thompson

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Bolling Air Force Base

Washington, D.C.

December 1995

PREFACE

Reports in this volume are numbered consecutively beginning with number 1. Each report is paginated with the report number followed by consecutive page numbers, e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3.

This document is one of a set of 16 volumes describing the 1995 AFOSR Summer Research Program. The following volumes comprise the set:

VOLUME	TITLE
1	Program Management Report
	Summer Faculty Research Program (SFRP) Reports
2A & 2B	Armstrong Laboratory
3A & 3B	Phillips Laboratory
4	Rome Laboratory
5A, 5B, & 5C	Wright Laboratory
6A & 6B	Arnold Engineering Development Center, Wilford Hall Medical Center and Air Logistics Centers
	Graduate Student Research Program (GSRP) Reports
7A & 7B	Armstrong Laboratory
8	Phillips Laboratory
9	Rome Laboratory
10A & 10B	Wright Laboratory
11	Arnold Engineering Development Center, Wilford Hall Medical Center and Air Logistics Centers
	High School Apprenticeship Program (HSAP) Reports
12A & 12B	Armstrong Laboratory
13	Phillips Laboratory
14	Rome Laboratory
15A&15B	Wright Laboratory
16	Arnold Engineering Development Center

ADVANCED C/C++ PROGRAMMING, GRAPHICS TECHNIQUES, AND NETSCAPE ENHANCED HYPERTEXT MARKUP LANGUAGE

Jonathan C. Bakert

Sauquoit Valley High School
Sauquoit, NY 13456

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

ADVANCED C/C++ PROGRAMMING, GRAPHICS TECHNIQUES, AND NETSCAPE ENHANCED HYPERTEXT MARKUP LANGUAGE

Jonathan C. Bakert
Sauquoit High School

Abstract

Numerous advanced C/C++ programming functions and methods were studied including overloaded functions, pointers to functions, and modularity techniques, as well as graphics techniques, such as image scaling, rotation, translation, double-buffering, palette manipulation, and ray-casting. In addition, the HyperText Markup Language (HTML) was learned in order to develop a "home page" accessible through the world-wide web, or WWW. Integrated into the HTML were clickable imagemaps and C code written to translate a decimal number (base 10) into a hexadecimal number (base 16).

ADVANCED C/C++ PROGRAMMING, GRAPHICS TECHNIQUES, AND NETSCAPE ENHANCED HYPERTEXT MARKUP LANGUAGE

Jonathan C. Bakert

Introduction

Programming languages, specifically C, tend to be the most efficient way of manipulating a computer to preform the necessary tasks to get a job done. This is doubly true when speaking of graphical techniques such as image scaling, rotation, translation, double-buffering, and ray-casting. Such manipulations can be done by assembling the necessary C functions, and then doing minor rewrites depending on your current situation. Double buffering is a graphical technique used to create smooth flicker-free animation as seen in many applications and is implemented by having two memory "pages" and switching between them as desired. Ray casting is used to create a 3D scene and consists of casting out a ray of light (which could be mentioned as a photon, or light having the properties of a particle) to its destination. This technique is very similar to the technique of ray-tracing, a graphical technique used to simulate the reflections of light and other very mathematically complex and CPU intensive operations.

Also, the hypertext markup language (HTML) was studied and the Netscape enhanced features (such as tables and centered menus) were implemented. Shell scripts were also incorporated so C programs could be utilized through the HTML script seamlessly by the user.

Methodology

This year, the C programming at its best was studied. Up until now, all of the programming techniques used were nothing drastically different than other languages, and could probably be done (perhaps faster) in languages such as (dare I say it?) BASIC. However, powerful programming techniques, such as multiple levels of indirection (which in C is described as a double pointer) can be used to move large amounts of data around simply by modifying a pointer (or a variable containing the address of another variable) to that data. Another powerful tool is actually being able to have a pointer to a function.

For example, in a structure you can contain code to be a pointer to a function. Then, if you want to modify one variable so that when it is called it references a different function you simply do something like the following code:

```
// Demonstration program for showing the usefulness of pointers to functions
#include <stdio.h>
#include <stdlib.h>

// This is the prototype of the function we want to have a pointer to and
// another function we will use to demonstrate the usefulness of this
// technique

float action(double x, double y, double z);
float action_2(double x, double y, double z);

// Now we can declare a pointer

float (*p)(double x, double y, double z);

main()
   {
   float status;
   // Declare p to point to the action function

   p = action;

   // Now we can call this function just by doing this

   status = p(1245, 1244, 987);

   // And you can change what function p is by just doing this

   p = action_2;

   status = p(1245, 1244, 987);

   // Now, p will call the action_2 function, and not the action function
   // So, when the loop continues, it will call the action_2 function but
   // still use the name p().  In large programs, this can be indispensable.

   }
```

However, it is possible to make an easily overlooked mistake when writing such code. For example, you may be wondering why you can't just do something like this to declare a pointer to a function (as you would with any other variable):

```
// Incorrect code to declare a pointer to a function

float *action(double x, double y, double z);
```

At first glance this may look correct, but in truth, it's not doing what we want. What this does is declare a function

which RETURNS a pointer to a floating point number. Since the * symbol (or the "pointer" symbol in C) has a lower level of preference than the parenthesis, the function is evaluated as returning a pointer but not having a pointer point to it.

```
// Sample code to declare a pointer to a function

float (*p)(double x, double y, double z);
```

The problem is corrected by adding the parenthesis around the *p.

Another programming approach which was studied was the usefulness of overloaded functions. Overloaded functions is a C++ programming technique which enables you to perform the following:

```
// Sample code to demonstrate function overloading

int mult(int x, int y);
float mult(float x, float y);
double mult(double x, double y);

int mult(int x, int y)
   {
   return x*y;
   }

float mult(float x, float y)
   {
   return x*y;
   }

double mult(double x, double y)
   {
   return x*y;
   }
```

Then, when you want to multiply two numbers together, C++ will automatically determine which function to choose depending on what arguments you give the function. That way, you don't need to make a different function for a different kind of variable such as:

```
int mult_int(int x, int y);

float mult_float(float x, float y);

double mult_double(double x, double y);
```

The next area studied was C and its uses in graphical programming. At first glance, all of the C programming techniques used to write graphics code may seem very daunting, but after closer inspection it becomes apparent that the "reusability" of such functions is incredibly high. For example, once a routine to write a bitmap scalar is written, only a

few small changes have to be made in the code to have it usable by another program. And, if the function was written well, no changes to the code may be necessary. You can simply supply different arguments and have a completely new function. Discounting possible optimizations, it's completely reusable.

Before you can fully get into graphics on the standard PC VGA graphics card, or any other system for that matter, you have to know how video palettes are organized. The typical VGA palette is composed of 256 different Digital to Analog Converter registers or DAC registers used to store the red, green, and blue values of the appropriate colors. Since the monitor is an analog device and the RGB components are digital values, the DAC must take the digital values and convert them into an analog signal. This signal is then sent to the monitor and translates into the image you see on your screen. The registers can have values from 0 to 255 if the card is using 8-bit values, or 0 to 63 if you are using 6-bit values, which is what the standard PC VGA card is composed of as compared to XWindows colormaps.

Since each color in the VGA palette is composed of 3 different bytes for each of the colors you may think that this would total up to 24-bits of color. This would logically leave you to believe the following:

8 bits = 2^8 = 256

Thus, there are 256 different possibilities for this byte. Then, there are 3 bytes, so:

256^3 = 16,777,216

This would mean that there are roughly 16.7 million colors available to the standard VGA card.

Uh, no.

Isn't that what these new SVGA cards are? 16.7 million colors? I thought VGA cards only have 256 colors? That too is partially correct. But, I've been meandering around the truth for a while. Here is how it really is.

The standard VGA card has 256 colors registers. These are the number of colors that can be DISPLAYED ON THE SCREEN AT ONE TIME. The amount of colors AVAILABLE on a VGA card are quite different. As I said before, there are 3 bytes in the 256 VGA DAC registers, however, ONLY 6-BITS OF THE 8 BITS ARE USED. So, that leaves us with this:

6 bits = 2^6

Thus, there are 64, not 256 different possibilities for this byte. Then, there are 3 bytes, so:

64^3 = 262,144

Thus, there are actually 262,144 colors available to a VGA card. But remember, only 256 of these are available to the VGA card at one time, greatly limiting the number of colors on the screen at one time. And, if you change one of these registers, all of those occurrences of that color on the screen will be changed to that color. This accounts for the changes in color on a screen when a colormap, or palette, is changed, and the entire screen becomes bizarre looking. And example of RGB values of various colors is:

```
245 245 220          beige
245 222 179          wheat
244 164  96          SandyBrown
210 180 140          tan
210 105  30          chocolate
178  34  34          firebrick
165  42  42          brown
233 150 122          DarkSalmon
```

The 3 integers (technically characters since they have a size of one byte since their value never exceeds 255) stand for red, green, blue respectively.

Once video palettes were understood, the various video modes available were examined. The two most popular video modes available on the standard PC VGA card with at least 256 kilobytes of memory are modes 13h and mode X. These video modes are most often used when fast graphics are required for techniques such as double buffering or bitmaps scaling and translation. The difference between the two video modes is only slight. Both modes support 256 colors, however mode 13h, unlike X, supports a 320x200 resolution while mode X is usually a 320x240 resolution. The second change is that mode 13h's memory is stored linearly. For example, the starting address is A000:0000, and ends with the last byte of memory. Mode X's memory is stored in pages. To reference mode X faster, various graphics packages "reshape" mode X to make it more manageable. It's a good idea to start in 13h and then progress to X once you have learned the intricacies of graphics programming.

In order to take advantage of the colormap, a program was written to expand the amount of colors in the Vogle graphics library. Although the library was sufficient for drawing 3-dimensional shapes, it lacked a lot of flare when it came to the topic of colors. Previously, the `color()` function in the Vogle library could only access about 8 colors. With the expanded `n_color()` function, Vogle now has access to over 600 colors located in a file on disk called `RBG.TXT`. By calling `n_color()` with a name from the `RGB.TXT` file, you can change the color to one of these, and theoretically as many colors as memory will allow.

The `n_color()` function is show here:

```
// Code for the n_color, expanded color function

int n_color(char *color_name)
   {
   int counter;

   for(counter = 0; counter <= num_colors; counter++)
      {
      if((strcasecmp(color_name, new_color[counter].name)) == 0)
         {
         mapcolor(8, new_color[counter].red, new_color[counter].green, new_color[counter].blue);
         color(8);
         }
      }
   }
```

To call this function, you simply enter the following line into your code in replace of the previous `color()` function:

```
n_color("darksalmon");
```

The `n_color()` function will automatically scan the `new_color[]` array and find the appropriate RGB values from the `RGB.TXT` text file. An example of the `RGB.TXT` file is:

```
245 245 220            beige
245 222 179            wheat
244 164  96            SandyBrown
210 180 140            tan
210 105  30            chocolate
178  34  34            firebrick
165  42  42            brown
233 150 122            DarkSalmon
```

Furthermore, if you want to add your own color, simply add your own 3 digits to this list, and supply it a unique name. Note that this function is NOT case sensitive, so if you put Brown or BrOwn, it's all the same.

Before you can use the `n_color()` function, you must call `init_n_colors()`. This will load the RGB values, and their corresponding name, into an array. An example of the `init_n_colors()` code is:

```
/* n_color: A function to use with the VOGLE graphics library to increase
 * the amount of available colors to over 600.  Must be used within a
 * VOGLE program unless the function MAPCOLOR is changed to something
 * which coincides with your library.
 *
 * Usage:
 *
 *      n_color("COLOR");
 *
 *      Where COLOR is a color name from the RGB color list in rgb.txt
```

```
 *      This function is not case senstiive
*
 *      REALLY WIZ-BANG IMPORTANT: You must call the function init_n_colors();
 *      before using the n_color function.  You may also add your own colors
 *      by modifying the rgb.txt file in the shown format.
 */
#define MAX_COLORS 800

typedef struct colors_header_type
       {
       int red, green, blue;
       char name[50];
       } colors_header, *colors_header_ptr;

colors_header new_color[MAX_COLORS];
int num_colors;

int init_n_colors(void)
   {
   FILE *fp;
   char temp[30], temp2[30], temp3[30];

   if((fp = fopen("rgb.txt", "rb")) == NULL)
      return(0);

   while(1)
      {
      fscanf(fp, "%s %s %s %s", temp, temp2, temp3, new_color[num_colors].name);

      if(feof(fp))
         break;

      new_color[num_colors].red = atoi(temp);
      new_color[num_colors].green = atoi(temp2);
      new_color[num_colors].blue = atoi(temp3);

      num_colors++;
      }
   fclose(fp);
   return(1);
   }
```

There is a drawback to this function however. If you're doing anything involving animation, and require a color far down on the RGB text list, you will notice a significant lag as the string entered is compared to the names in the RGB text list. However, since this is rarely done with this package, the current function is sufficient and very usable.

The next ideas explored were graphics concepts such as bitmap scaling, rotation, translation, double-buffering and ray-casting.

The idea of bitmap manipulation, in techniques such as scaling, rotation, and translation is not an entirely new one. It has been scene in numerous software packages where graphical manipulation is a must, such as image viewers, and

virtual reality software.

The first kind of bitmap technique is scaling. To scale the image correctly, you must find out how many times to replicate (or remove) a pixel of a certain color depending on its scale factor. For instance, take the simple manipulation of doubling the size of a bitmap:

Original data file for bitmap:

```
12   14   14   12   12   14   15   2   2   6
```

Original data file after a doubling transformation:

```
12   12   14   14   14   14   12   12   12   12   14   14   15   15   2   2   2   2   6   6
```

This results in the previous image being doubled simply by duplicating each byte of data from the current palette . As you can imagine, this technique works very well for manipulating objects with easy numbers such as 2 or 3. You simply replicate each byte as many times as you are increasing the size of the object. But, this certainly wouldn't give us a realistic, and believable scrolling of an object. Your world doesn't move in integer multiplication so why should your computer simulation. Well, it doesn't have to, but you already knew that.

Basically, the technique for manipulating an object so that it can scroll smoothly towards you is a matter of computing a scale factor which to base the replication of the RGB color values on. Since you want to have as realistic a scroll as possible, you'll need more than integer precision. But, this technique also has it's drawbacks. Since there is a finite amount of pixels, a transformation such as this is going to have its drawbacks:

Original bitmap data:

```
12   14   14   12   12   14   15   2   2   6
```

Bitmap data after undergoing a non-integer scaling transformation:

```
12   14   14   14  12   12   12  14   15   2   2   2   6
```

As you can see, although the image is actually scaled, its proportions are still not quite correct. But the problem lies in the ability to not have "half" a pixel. How exactly could you represent the first 12, for example, in this integer list? Well, you can't - at least not easily. This loss of precision is what accounts for the uneven scrolling of such bitmaped objects as it may become apparent in simulations or in the resizing of bitmap data.

The next facet of bitmap manipulation is rotation. Fast bitmap rotation without an apparent amount of loss is still

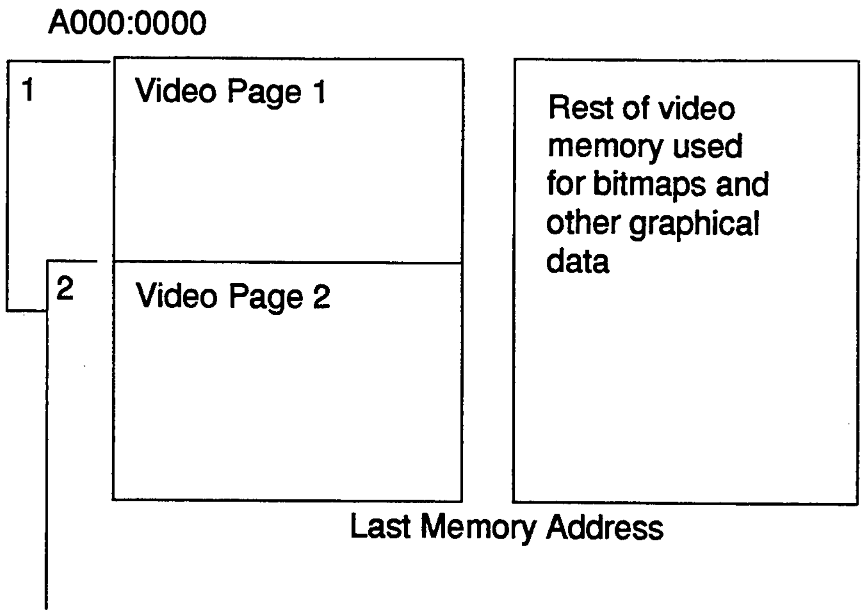
an area of study. Usually, when a bitmap is rotated, each of its pixels must have their new position computed using trigonometric functions. Of course, this results in the image degradation that is apparent with scaling. Usually, images will be rotated using a separate graphics package, stored, and then accessed from memory instead of being computed on the fly. However, this technique can only access as many images as you have saved, so on-the-fly mathematical rotation can be beneficial if speed is not of the essence (or you happen to have a quad-processor Cray handy, whichever may be most likely).

Bitmap translation, however, is perhaps the easiest of the different graphical techniques here. The only thing required is an initial offset value from the current position (or, if you don't prefer to work with relative coordinates, you can choose a different coordinate system, it's all the same) and then to move the appropriate amount of pixels. However, if the image is transposed over a background (which they usually are) updating the background must be taken into account. To solve this problem, the programmer must simply rewrite the background BEFORE he draws the bitmap. But, this has the effect of slowing down the cycle. A faster, but a bit more complex approach is to redraw only a small part of the background of your simulation which the bitmap previously covered. That way, a much smaller portion of the screen has to be redraw, and there is a considerable speed increase, even after taking into account of the overhead of the math to compute the area to redraw.

When programmers want flicker-free graphics in their simulation programming, they usually employ a technique called double-buffering. Double buffering is the act of having two pages of video memory allocated - one for what is displayed ON SCREEN and another with what is OFF SCREEN. By drawing your images in off-screen video memory and switching a pointer from one region of video memory to the other when necessary, almost flicker-free programming can be created.

The "almost" is derived from the problem with the vertical retrace. In non-interlaced monitors, the electron "gun" begins at the upper left hand corner of the screen and begins drawing horizontal lines, one by one, until it reaches the lower right hand corner of the screen. Then, when it reaches this position, it RETRACES back to the upper left hand corner of the screen. This brief moment when the electron gun resets is known as the vertical retrace. In order to have TOTALLY flicker free animation, you must have your graphics synch in with this. There is a way to do it.

First of all, here's a diagram on how to organize a 320 x 200 resolution with two pages:



A pointer to video memory will change from these two points for each frame the image needs to be updated.

On the VGA card, in the third bit of register 0x3DA is a bit which reports if the card is retracing, or not. When the bit is 1, there's a retrace in progress, if it's 0, there is no retrace in progress. It's optimal to transfer your pointer from the visual page to the hidden page the INSTANT the vertical retrace is over so you can avoid getting the flicker associated with starting to draw an image when the retrace is almost about to occur. So, by monitoring this particular register, you can have your program only redraw when it's optimal. This is how you can accomplish flicker free animation.

Lastly, the concept of ray-casting, an offshoot of ray-tracing, was studied. Ray casting is the technique of casting out artificial "rays", or photons (which is light as represented as a particle) until they reach a boundary, such as a wall, or hangar-bay, etc. Then, the distance the photon has travelled is computed, and the proper image is displayed on screen. The amount of rays which must be cast out is actually a segment of the field-of-view (FOV) of the observer. If the FOV of the observer is 60 degrees, than within that 60 degrees an amount of rays equal to the horizontal resolution of the screen, must be cast out. This is to render simple scenes, such as a room with rectangular walls, and no odd-shaped facets. More complex ray-casting engines must be designed to incorporate a more complex 3D environment.

The last topic studied during the course of the summer was the HyperText Markup Language, or HTML. HTML is the standard scripting language used on such internet browsers as XMosaic and the now popular Netscape. In fact, Netscape even has an extension of the HTML language it so modestly calls Netscape Enhanced. Here is an example of the actual HTML script that comprises my home page. I'll discuss unique sections as they appear through the document.

```
<HTML>
<BASE HREF="http://erda.rl.af.mil/~jon/">

<TITLE>Jon's Homepage</TITLE>

<HEAD>
<CENTER>
<TABLE border=6 width=40%>
<TD><H1>Jon's Homepage</H1></TD>
</TABLE>
</CENTER>
</HEAD>
```

There are a few unique "tags", as they are called in HTML, in the above text. An example of a tag is something such as <HTML> or <CENTER>. Thus, in the above code, <HTML>, <BASE>, <TITLE>, <CENTER>, <TABLE>, and <TD> are all legal tags in HTML.

When you want to end a tag, you place a forward slash (/) in front of the name to show that whatever you wanted done should be stopped. For example, the <HEAD> tag tells the browser that this is the header of the document. When you see the </HEAD> tag you know that the header has ended. The same applies for <CENTER> and complex tags such as <TABLE>. When you see </TABLE> it means the table has ended, and when you see </CENTER> it signifies that the </CENTER> tag has ended. Thus, it creates a centered table.

```
<P>
<A HREF="/jon-cgi-bin/imagemap/wolfatnight">
<IMG WIDTH=535 HEIGHT=200 SRC="gif/wolfatnight.gif" ISMAP></A>

<P>
<IMG SRC="gif/hr.gif">

<H3>
<IMG ALIGN=CENTER SRC="gif/netscpen.gif">This page is Netscape ENHANCED!
</H3>

<table border=7 width="100%"><tr align=center>
<td><h3>Welcome to Jon Bakert's WWW Page!  Devoted to RPG's, LRPG's, Music
and the Mobious spheroid constant!</h3>
</td><td>
If you don't see something you would like on this page, just mail me my selecting
my address atthe bottom of this page (Netscape only) and I'll add it
</td>
</table>

<P>
<IMG SRC="gif/hr.gif">
<UL>
<H2>
<IMG ALIGN=MIDDLE SRC="gif/eye.gif">
C programming</H2>
```

```
<UL>
<H4>
<IMG SRC="/Images/red-ball.gif">
A useful <A HREF="c/num2hex.c>decimal to hexadecimal</A> converter
which you can
<A HREF="n2h.html">use</A>!</H4>
<IMG SRC="/Images/red-ball.gif">
A not so useful <A HREF="c/ftest.c>test program</A> I wrote.  I don't
have anything else!
</UL>
</UL>

<IMG SRC="gif/hr.gif">
```

A few new HTML tags are now seen in this text, specifically the most useful of them all, the `<IMG>` tag. This tag allows you to insert graphics into your documents making your page all the more attractive to look at. Also, an advanced technique, imagemaps, was incorporated into the page. By using clickable images, you can specify the browser to go to a different "link" (which can be anything from another HTML document to an FTP site) depending on where the user clicked in an image. An example of such a map file is shown below:

```
default http://erda.rl.af.mil/~jon/index.html
rect http://erda.rl.af.mil/~jon/evil.html 70,150 200,200
```

The "default" keyword used here is the link the browser will use if it doesn't find a match depending on the coordinates the user clicked on in the image. The `rect` keyword stands for rectangle, and the browser scans the line for the four integers which make up the rectangles x and y coordinates. If the user's click was within these confines, it will go to the link `http://erda.rl.af.mil/~jon/evil.html`. Valid keywords are also `circle` and even `polygon` (where you can have over 1000 facets). Now back to the HTML code...

```
</BODY>
<IMG SRC="gif/hr.gif">
<P>
<IMG SRC="/Images/lock.gif"> indicates items under construction.
<P>
This page has been accessed <A HREF="http://www.forsmark.uu.se/counter.html"><IMG
SRC="http://www.forsmark.uu.se/cgi-bin/counter/jon-hsap"></A> times!  (I may
     start to charge, so watch out America!)

<HR>
<P>
Mail Jon at: <A HREF="mailto:jon@erda.rl.af.mil">
<IMG SRC="/Images/mail.gif">
jon@erda.rl.af.mil</A>
</HTML>
```

As you can see, the `</BODY>` tag is used to show the end of the body of the HTML code. After that is an area that I use if any *Netscape* users want to mail me. By creating an anchor (using the `<A>` tag) and inserting the "mailto:" keyword, a simple click on either the mail icon (the IMG tag which references mail.gif, an image file) or my address ("`jon@erda.rl.af.mil`") will bring up a window which contains the necessary forms to send me a message. And, if you've been reading this and wondering what's in `evil.html`, well, I'll show you.

Once the user has clicked on the right spot on the bitmap, the browser will determine if the user clicked on the wolf in the wolfatnight picture. If the user did, you're in for a surprise. The following code is executed many, many times.

```
<body TYPE=fadein bgcolor="#ffffff">
<body bgcolor="#f0f0f0">
<body bgcolor="#efefef">
<body bgcolor="#eaeaea">
<body bgcolor="#e5e5e5">
<body bgcolor="#e0e0e0">
<body bgcolor="#dfdfdf">
<body bgcolor="#dadada">
<body bgcolor="#d5d5d5">
<body bgcolor="#d0d0d0">
<body bgcolor="#cfcfcf">
<body bgcolor="#cacaca">
<body bgcolor="#c5c5c5">
<body bgcolor="#c0c0c0">
<body bgcolor="#bfbfbf">
<body bgcolor="#bababa">
<body bgcolor="#b5b5b5">
<body bgcolor="#b0b0b0">
<body bgcolor="#afafaf">
<body bgcolor="#aaaaaa">
<body bgcolor="#a5a5a5">
<body bgcolor="#a0a0a0">

[bit of code deleteted - I think you get the idea]
```

What this is doing is actually pretty silly. The six letters/digits are actually hexadecimal code which stand for the RGB values (see how this all ties in?) of the background color of the screen. The rest of the code goes on to decrement the 3 numbers until they reach 0. This makes the screen appear to flash. Then, at the end an image of Elvis Costello screaming is seen along with a message to the user! Isn't this just pure power?

Lastly, incorporated into the document is a link to a shell script that looks like the following:

```
#!/bin/sh

#PATH=/usr/local/bin:/usr/local/bin/ptybin:/usr/ucb:/bin:/usr/bin:/usr/etc
#export PATH

/home/vortex/httpd/cgi-bin/rl-post-query $CONTENT_LENGTH ${$}
```

```
TEMPFILE=/tmp/${$}

DECIMAL=`grep 'decimal =' $TEMPFILE | sed -e 's/decimal = //'`

HEX=`bin/n2h $DECIMAL`

echo '<!doctype html public \"-//W30//DTD W3 HTML 2.0//EN\">'
echo '<HTML><HEAD>'
echo '<TITLE>That Wacky Hex Number!</TITLE></HEAD>'
echo '<P>'
echo '<H2>Your decimal when translated to hex is:</H2>'
echo '<H4>'
echo $HEX
echo '</H4>'
echo '</HTML>'

rm -f $TEMPFILE
```

What this script does is execute the program `n2h` which is a decimal to hexadecimal converter I wrote. It does this by executing a program called rl-post-query and supplies it with the content length of the number typed in, and the Program ID (or PID) number to store it's information in. The next few lines rips apart the PID file, and sends the users number into the n2h program which then echoes out the answer (in Hex) and stores it in the shell variable HEX. The next few echo lines echo out the required HTML code which the browser translates along with the answer to the problem. Lastly, the temporary file is deleted so that there are not a lot of temporary files lying around.

Results

The results of this year's summer apprenticeship were great. So much more was learned about using the C, C++, and HTML that it is hard to believe I was getting by without this information. Furthermore, advanced graphical techniques, such as rotation, scaling, and translation are proving to be an invaluable asset when entering into a field which is utilizing programming techniques of a gradually more complex nature.

The outcome of all this work is a greater understanding of programming languages, a more firm, overall grasp on the concept of modularity, and a more diverse background in graphical routines which will prove invaluable in the direction which are society is headed. The HTML language is also proving to be a very beneficial course of study. Once you master the HTML language, the entire world can access your own "page" which contains your personal preferences, and possibly other peoples too! This is a perfect way to get acquainted with everyone's favorite buzz-word, the information super-highway, at an easy and no longer obscure method. Undoubtedly, programming in all of these languages will prove a boon to those who wish to stay in the fast lane of the technological era.

Conclusion

In summation, this summer was indeed well spent. After learning some of the new techniques presented, old C code can be reworked and improved using such techniques as linked lists, function overloading, and stricter modularity. The 3D graphical techniques which were learned are already showing their impact on society as this is written. Simulations such as virtual reality demonstrations, are already proving their popularity in both the technical and entertainment fields. The military and the consumer has something to gain from these extraordinary new ideas and implementations.

And, with the added skill of learning HTML, the world is now also not out of bounds for anyone. With the right server up and running, even those in Finland aren't out of touch. In fact, they're only a few seconds away with the high speed access the internet provides for us. The future ahead looks exciting, so the best advice I can give is to sit back and enjoy the ride!

References

1. Gruber, Diana. Action Arcade Adventure Set. Coriolis Group Books, 1994.

2. Davis, Stephen. C++ for Dummies. IDG Books Worldwide, Inc., 1994.

3. Peter Aitken & Bradley Jones. Teach Yourself C in 21 Days. SAMS Publishing, A Division of Prentice Hall Computer Publishing, 1992.

4. LaMothe, Ratcliff, Seminatore & Tyler. Tricks of the Game Programming Gurus. SAMS Publishing, A Division of Prentice Hall Computer Publishing, 1994.

THE PREPARATION OF MATERIAL FOR ACCESS ON THE WORLD-WIDE WEB

Daniel T. Brown

Sauquoit Valley High School
2601 Oneida Street
Sauquoit, NY 13456

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

THE PREPARATION OF MATERIAL FOR ACCESS ON THE WORLD-WIDE WEB

Daniel T. Brown
Sauquoit Valley High School

Abstract

Slideshows were prepared showing the differences that the Knowledge-Based Software Assistant (KBSA) could make in the software world. These slideshows were then converted from the Macintosh PowerPoint format into a PowerPoint application for Microsoft windows. This format was than printed as a postscript and transferred to a Sun computer. The postscript was then edited to be readable by the World-Wide Web server. Each individual slide was then given a ".gif" extension so that they would be readily available for use as inlined images in Hypertext Mark-Up Language (HTML). Many concepts of HTML 2.0 were also studied in-depth.

THE PREPARATION OF MATERIAL FOR ACCESS ON THE WORLD-WIDE WEB

Daniel T. Brown

Introduction

In today's world, where it seem as though everything is being put on a computer, the World-Wide Web, or Internet, is thriving. It is accessed millions of times a day for the purpose of gathering information. To help the public to better understand what goes on inside Rome Laboratory, the many branches have prepared data for placement on the Internet. Hopefully this data will clear up some of the questions that the public may have concerning the bearings of Rome Laboratory. However, due to their research, some branches haven't had the time to convert this data into a form readily usable by Internet browsers such as Mosaic or Netscape. As a result, these files had to be converted so that they could be recognized by the branch server and accessed freely. This was done mainly through the editing of each file's postscript and the use of several image editors. Some knowledge of HTML 2.0 was also applied.

Methodology

The aforementioned data was, in fact, a slide show presentation showing the development and benefits of applying the Knowledge-Based Software Assistant (KBSA) to the development of software. This slideshow was converted from a Macintosh PowerPoint application to one that is acceptable by the branch's Sun computer, which is the Internet server. This conversion was accomplished by the use of a 6100 PowerMacintosh because of its ability to translate to DOS applications. Once the slideshow was placed on a DOS formatted disk, it was translated to a PowerPoint application for Microsoft Windows. This file, once translated, was printed to a file in the form of postscript. Using the UNIX file transfer protocol, the entire slideshow was sent to

a remote console so that the editing may begin in a format easily recognizable by the Sun server (Mercury). The postscript of the whole slideshow was edited next. This was done by removing offending characters such as the "control D" (^D) characters in the script and inserting "%!" characters at the beginning and end of each script. These scripts, once edited were now fully readable by Mercury, the server mentioned above. Now the slides had to be viewed one at a time and saved separately. This was done using the Solaris Image Tool on a Sun computer much like Mercury. Once the entire slideshow had been broken down into its individual slides, they had to be further modified. These slides, as a result of the original slideshow, were still in a postscript format and were unable to be placed on the Internet in an easily accessible form. These slide show had to be changed into either a *.gif* (GIF) or *.jpg* (JPEG) format. Both of these formats can be found on the Internet, but it so happens that the GIF (Graphical InterFace) is the more widely accepted format, so it was decided that the slides would be converted into a GIF format by changing the extensions. The extensions were changed and the slides are now ready to be placed on the Internet for public access. These slides may be presented as either a complete slideshow or simply be placed on the Internet as separate inlined images through the use of some version of HTML (the current version is HTML 3.0, but HTML 2.0 is still in wide-spread use). Three versions of the slideshow were converted.

While converting these slideshows, I was also learning the HyperText Mark-up Language, or HTML, as it is commonly known. I worked exclusively on version 2.0, for version 3.0 isn't in wide-spread use yet. While studying HTML 2.0, I learned a great many things. For example, I can define the title, body, and text of a HTML document with great ease. These basic applications, and even the more complex ones are brought about through the use of *tags*. These tags are commands given in the format of a beginning tag and an ending tag. For example, the following would be an example of a document's title:

<title>A World-Wide Web Document Title</title>

As you can see, the command is placed in a set of < > to begin its application, and the application is terminated with the combination of </ >. The case of the letter is not important because for ease of use HTML was designed not to be case-sensitive. Some of these tags are listed below:

Common HTML Tags

<title>...</title>	Identifies the document's title on the Internet
<body>...</body>	Identifies the document's main body of text
...	Creates a header--large, bold type (h2-- h6 create less prominent headers)
<a href-"...">...	Creates a hypertext link to another document
<img src="...">	Places an in-lined image in the document.
<p>	Paragraph break. This is equivalent to a double-return
<i>...</i>	Creates italic type
...	Creates bold type
<code>...</code>	Creates a fixed-with type

The above tags are all that are really necessary to create a "bare-bones" HTML document. These, however, are just the basics. More advanced tags exist to add more to your document and to ease in access of materials. Most of these tags are not new, but are, in fact, simply extensions of the basic HTML tags.

One such example of an advanced tag would be the addition of the *align* option in the placement of an in-lined image. The default of such images is to place the bottom of the image on the same line as the text is placed on in the document. However, by placing the option *align=top*, or the *align=middle* option, you can adjust the positioning of the image. Of course, both will align the image with the text at the location specified. In-lined images are very versatile. Not only can they be used to make you document look more appealing, but they can serve as *linked images*, too. This means that by clicking on the image, it can take you to another location on the Internet.

These are handled the same way as hypertext is and behave in the same way. Hypertext will be discussed in more detail later on.

The ultimate use of an in-lined image would be the creation of an *imagemap*. These in-lined images are very special due to the fact that you can specify certain areas of the image to be linked to various locations all over the Internet. This is done by writing a script that is placed on your Internet server. Then a normal HTML document is created with the following exception: the characters **ismap** are placed after the image's source within the image tag. This results in the referral of the browser to the server to see in what area the map was clicked, and to what location the browser should go. This allows you to place several links in one location and not have to type all of the text needed for the use of regular hypertext. The areas of an imagemap can be specified in many different shapes. These have been listed below:

Defining Regions of an Imagemap

Circle	Specifies a circle using a centerpoint and a radius
Poly	Specifies a polygon with up to 100 vertices
Rect	Specifies a rectangle using two points.
Point	Specifies a point that will recognize a click in its general area

Hypertext is by far the most common thing used throughout the Internet. Hypertext is simply text that has been linked to another location on the Internet. This is done through the creation of a link. This is accomplished by using the proper HTML tag and specifying the correct Universal Resource Locator (URL). Therefore a link would look something like this in a document's script:

click here

This would result in the words "click here" being highlighted and click-sensitive, so that a click

on the text would take you to the location "the linked document's URL." Such applications make the cross-referencing of material extremely easy. This is accomplished by placing linked hypertext throughout your document to provide quick access to related information. This may result in a greater number of visitors on your Internet site as a result of speed and ease of access. Hypertext can also be used to supplement the use of linked images and imagemaps. This is because some browsers on the Internet may not be able to handle some formats of images (.gif, or .jpg). This handicap could result in the inability to access certain links presented in the form of linked images. The addition of alternate hypertext would allow these users to access the same information with minimal hassle. Once again, this simple action could increase the number of visitors to your pages. As you can see, hypertext is a staple of the Internet for many locations--many would be rendered useless without it.

Various image formats are available for use on the Internet. The most common are the GIF and JPEG formats. The X11 bitmap is also available, but is only capable of black and white graphics. Some browsers will only work with certain formats. For example, the Netscape browser will only accept JPEG formats when loading a local file. HTML 2.0 is capable of only handling two different formats for the use of graphics. These two are the GIF and X11 bitmap formats. Although this is not a major limitation, it can slow a page's creation if you must spawn a viewer to alter its format. As a rule, its best to make sure that you have all of your planned images stored in the correct format on your server prior to your release of your page on the Internet. This is to make sure that as soon as it's placed on the World-Wide Web, all visitors can access the graphic you have placed there. This is simply a courtesy to your visitors.

Many special effects are available to you in HTML 2.0. Some of these effects are the following: italic text, bold text, emphasis (another version of the italics effect), strong emphasis (similar to the bold text effect), blinking text, and other that are too numerous to mention. These simply add

elements that will make your page more appealing to the eye. Some, such as the blinking text, can be combined with hypertext to draw your attention to a certain spot on your page. All in all, these effects allow one to personalize a homepage and make information more eye-catching.

The traditional ways of simply listing information have not been forgotten. Lists have not been left out of the HTML, and each has its own separate tag. Bulleted list are available and present the information proceeded by dots or "bullets." Ordered lists present lists that are numbered in the order that they are entered into the HTML document script. Description lists make a format similar to a dictionary available to create lists of items that will require description for each entry. The first two types of lists use an identical tag that simply identifies the item to be listed. How these items are listed depends upon the list tag that led off.

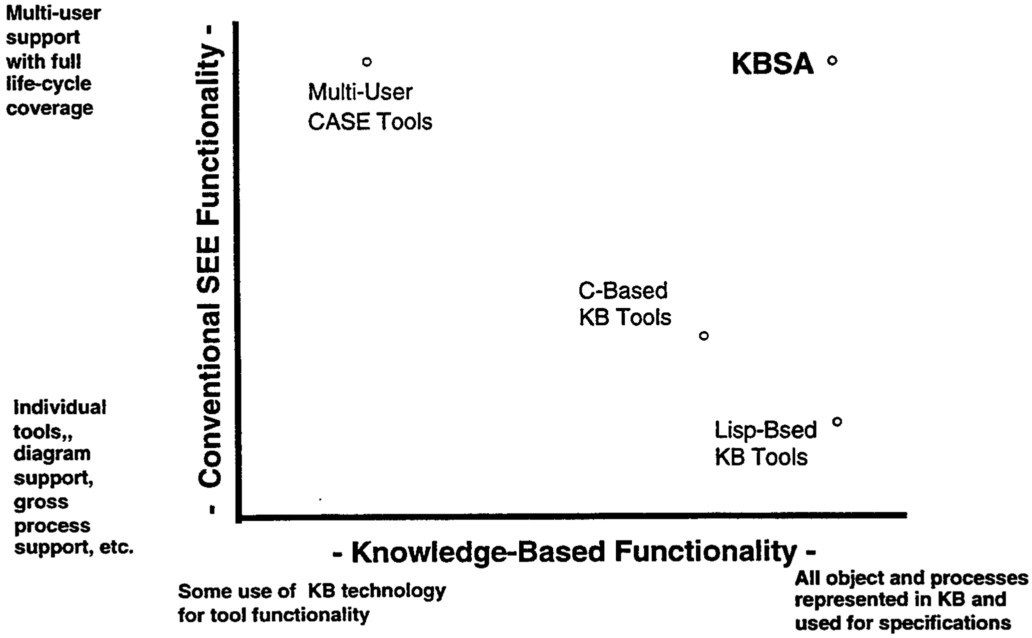
HTML Tags For Lists

	Creates an unordered, or bulleted list
	Creates an ordered, or numbered list
	Identifies the item to be placed in an unordered or an ordered list
<dl> </dl>	Creates a description list
<dt>	Identifies the description's title
<dd>	Identifies the actual description

In conclusion, HTML, in any form or version can be used to make information easily accessible to anyone who has access to a browser capable of accessing the Internet. As a result of such easy access, one is able to find the needed information and utilize it more efficiently, greatly improving productivity.

A Sample Slide For Conversion

Conventional S/W Technology vs. KBSA Technology



A Sample HTML 2.0 Document

```
<html>
<title>HTML 2.0 Document</title>
<body>
<h1>Sample HTML Document</h1>

This is the first paragraph of this sample HTML 2.0 Document.  It is going to be very short--only
long enough to illustrate the structure of a HTML script.<P>

This would be the second paragraph.  It contains an in-lined image with the text aligned along its
top border.<img align=top src="picture.gif"><P>

The third paragraph has two descriptions:<br>

<dl>
<dt>Title one
<dd>Description one
<dt>Title two
<dd>Description two
</dl><P>

Here's an Imagemap to help you find your way around:<img src="sample_map.gif" ismap><P>

Some lists are also included.
<ol>
```

item 1
item 2
item 3
<P>

item
item
item
<P>

My document is drawing to a close. Feel free to mail me at <address>brown@ai.rl.af.mil
</address>
</body>
</html>

Resources

"A Beginner's Guide to HTML."

http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Demo/html-primer.html

"Otmar's List of HTML Tags."

http://www.dec.uchile.cl/~amatico/mvasquez/tags.html

"The Bare Bones Guide to HTML."

http://www.access.digex.net/~werbach/barebone.html

"A Beginner's Guide to HTML."

http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html

"HTML Quick Reference."

http://www.ie.cuhk.hk/mirror/HTMLQuickRef.html

"Graphical Information Map Tutorial."

http://wintermute.ncsa.uiuc.edu:8080/map-tutorial/image-maps.html

"Clickable Imagemaps." http://gnn.com/gnn/bus/ora/features/miis/index.html

"Hypertext Mark-up Language 2.0"

http://www.w3.org/hypertext/WWW/MarkUp/html-spec/html-spec.txt

HIGH FREQUENCY CHANNEL PROBE EXPERIMENT AT MID-LATITUDES

Elaine Y. Chen

Lincoln-Sudbury Regional High School
390 Lincoln Road
Sudbury, MA 01776

Final Report for:
High School Apprenticeship Program
1995 Summer Research Program

Sponsored by:
Rome Laboratory
Hanscom Air Force Base, Massachusetts

and

Air Force Office of Scientific Research
Bolling Air Force Base, Washington DC

August 1995

HIGH FREQUENCY CHANNEL PROBE EXPERIMENT AT MID-LATITUDES

Elaine Y. Chen
Lincoln-Sudbury Regional High School

Abstract

High frequency signals are important in long range communications, broadcasting, over-the-horizon surveillance, and ionospheric sounding. However, these signals can be degraded by ionospheric irregularities. These irregularities mainly occur in the auroral zones and equatorial regions, but there still may be significant effects of ionospheric irregularities on HF signals in mid-latitude regions. An experiment was designed to find the relative contribution of clutter of the direct paths of HF signals at mid-latitude compared to the off-great-circle paths. Two transmitters in Florida sent signals to a receiver placed in Chesapeake, Virginia. FM/CW waveforms with sweep bandwidths of 25.0 kHz, 12.5 kHz, and 4.17 kHz were used in the frequency range of 6 MHz to 12 MHz. Data were collected from 21:00 to 07:00 UT during January 11-14, 1995. The results were inconclusive about whether off-axis scattering is a significant contributor in degrading the quality of HF signals.

HIGH FREQUENCY CHANNEL PROBE EXPERIMENT AT MID-LATITUDES

Elaine Y. Chen

Introduction

The ionosphere is the region of the earth's upper atmosphere that consists of several overlapping ionized layers. Ionization of these layers depends primarily on the sun and its activity. The ionosphere is a dynamic system controlled by many parameters, including time, geographical location, and certain solar-related ionospheric disturbances. It is divided into four broad regions: D, E, F, and topside. The F layer, which is the region of primary interest to radio communications, can be further divided into the F_1 and F_2 regions. Of these two layers, the F_2 layer, which has an altitude of about 200 to 400 kilometers at mid-latitudes, is the most important layer in terms of high frequency (HF) radio propagation [McNamara, 1985]. The HF portion of the electromagnetic spectrum is between 3 and 30 MHz. Signals of these frequencies can be reflected by the F_2 layer of the ionosphere, making HF signals important in long range communications, broadcasting, over-the-horizon surveillance, and ionospheric sounding. However, HF propagation can be affected by small scale ionization irregularities in the ionosphere. Although these irregularities seem to be in every level of the ionosphere, they are predominantly in the F layer [Aarons and Basu, 1985; McNamara, 1985]. When HF signals traverse ionospheric irregularities, they experience signal fluctuations. These signal fluctuations mainly occur in the polar, auroral zones and equatorial regions. While scintillation activity in these regions is more intense than at mid-latitudes, there still may be significant effects from ionospheric irregularities on HF radio propagation in the mid-latitude regions.

Discussion of Problem

Understanding more about the mid-latitude ionosphere in relation to HF propagation could help maximize the benefits of HF propagation. The effect of the ionosphere, particularly the effect of off-axis

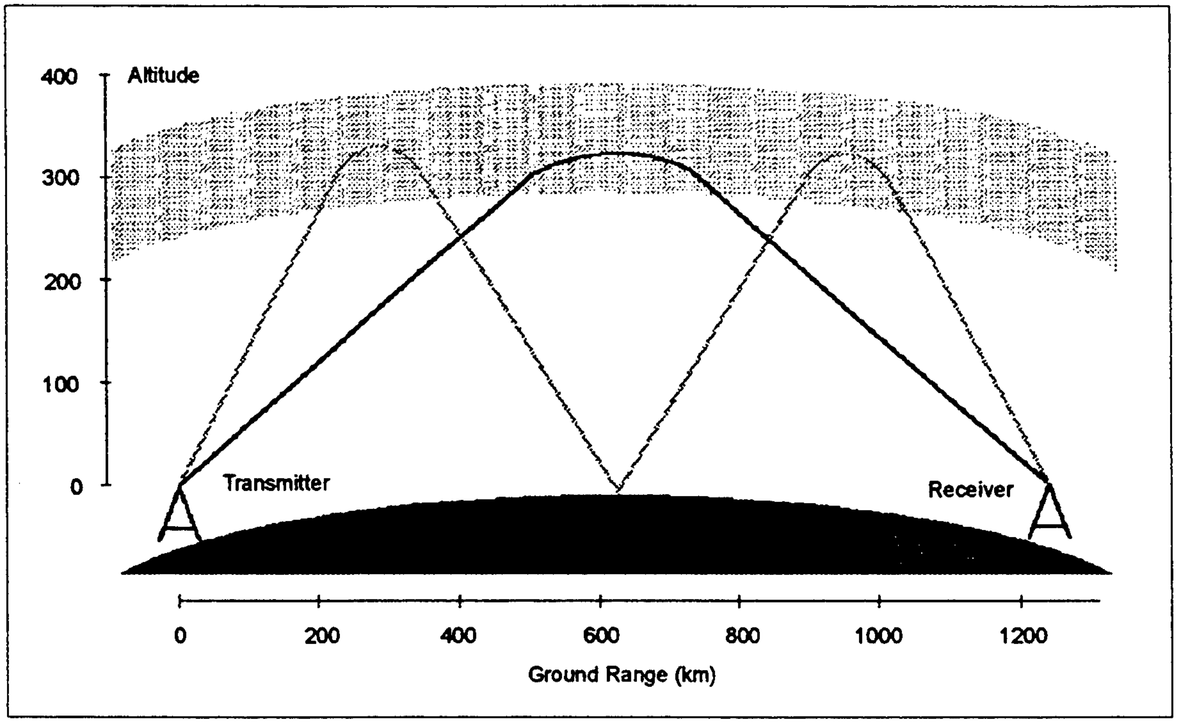


Figure 3-1. Ray paths for one-hop and two-hop propagation modes.

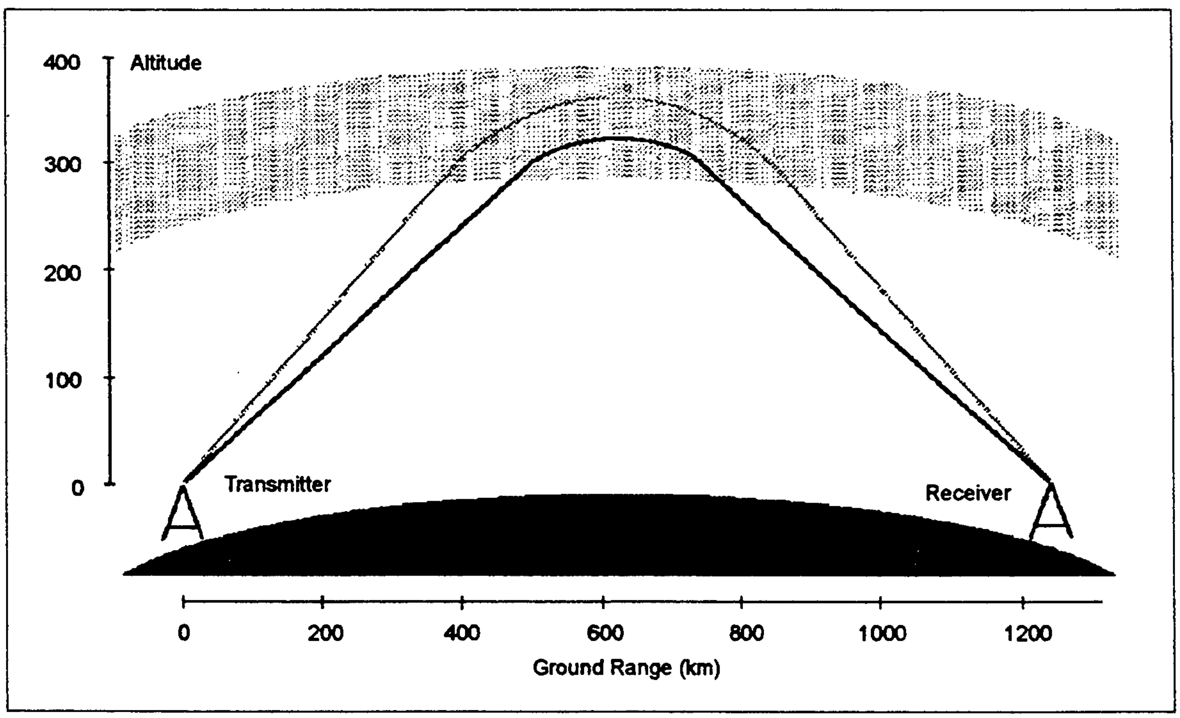


Figure 3-2. Ray paths for high ray and low ray propagation modes.

ionospheric clutter, in reducing HF signal coherence might be determined by measuring the HF channel scattering function. In doing so, the various possible propagation modes must be taken into account. For example, a signal could undergo a one-hop or a two-hop propagation mode (Figure 3-1). In this case, the receiver would detect two signals since it would take the two-hop signal longer to reach the receiver. The possibility of more than two hops is extremely unlikely considering the short ground range of 1,250 km. Likewise, the high ray/low ray propagation mode would also produce two signals (Figure 3-2). The high ray, which enters the ionosphere at a greater angle, would penetrate higher than the low ray before it can be refracted back toward the receiver. In addition, there is the possibility of off-axis scattering sources and mid-path phase modulation sources [Franchi and Tichovolsky, 1989] (Figure 3-3). Because the experiment focuses on the latter case, attempts were made while collecting the data to reduce the possibilities of two-hop and high ray propagation.

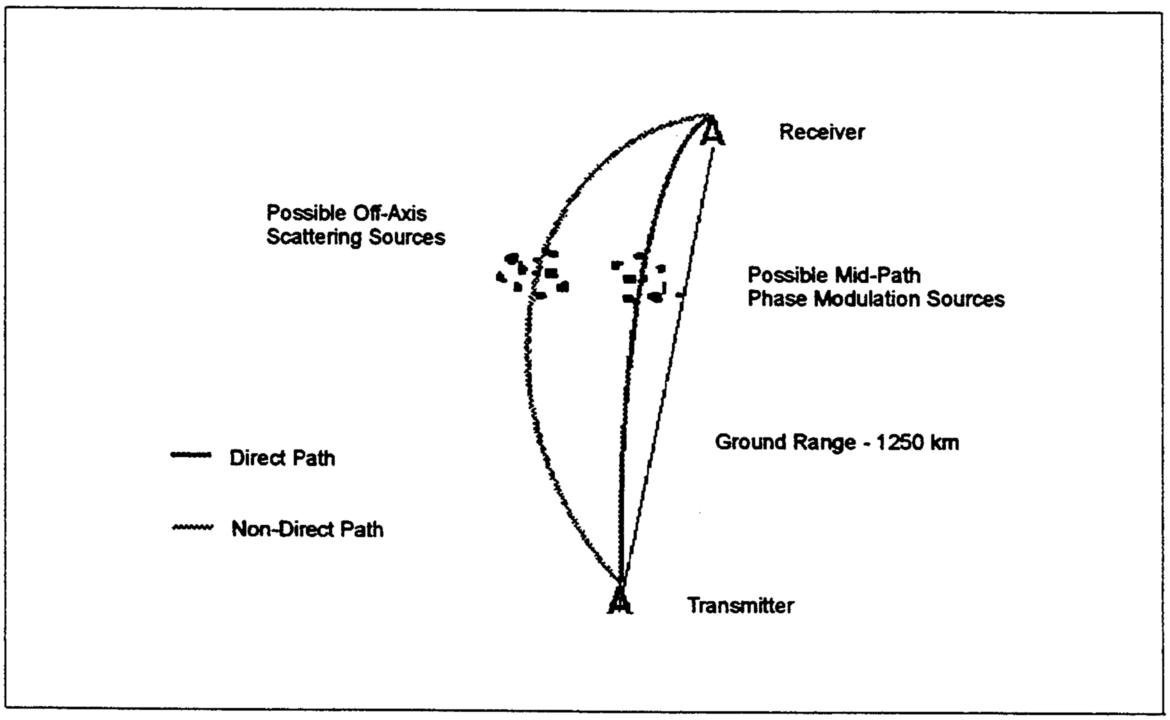


Figure 3-3. Ray paths for direct ray and non-direct ray propagation modes.

By determining which paths the signals took, it would be possible to observe the effect of off-axis clutter in degrading the signal quality of the HF signals. This will help in developing mitigation techniques to compensate for the distortions caused by off-axis clutter for HF communication systems operating at mid-latitudes. Also, it is a step towards the understanding of HF propagation and being able to accurately predict the HF environment.

Methodology

Two transmitters were placed at Homestead Air Force Base in Florida at a latitude of 25° 29' 38.4'' and longitude of 80° 23' 37.8''. One of these transmitters had an omnidirectional folded dipole antenna and an output power of 1 kilowatt. The other had a sloping 'V' Antenna and 40 watts of output power. These transmitters were used in conjunction with a receiver with an omnidirectional monopole antenna located at a latitude of 36° 33' 49.2'' and longitude of 76° 14' 54.0'' in Chesapeake, Virginia. This placement created a receiver-transmitter ground range of 1,250 km.

The data were collected from 21:00 to 07:00 UT during January 11-14, 1995. In collecting the data, an FM/CW (frequency modulated/continuous wave) waveform with three different sweep bandwidths were used. Two of the sweep bandwidths were 25.0 kHz and 12.5 kHz. The third was the relocatable over-the-horizon radar (ROTHR) waveform code 78, which has a sweep bandwidth of 4.17 kHz and a 96 msec waveform repetition period. For all three sweep bandwidths, a coherent integration time (CIT) of 6.1 seconds and a frequency range of 6 MHz to 12 MHz was used. The operating frequency was changed regularly to stay near the maximum usable frequency (MUF), which would minimize the possibility of other propagation modes such as two-hop and high ray. The MUF was determined with the aid of backscatter ionograms that were taken about every fifteen minutes for the duration of the experiment.

All of the collected data was processed, graphed, and studied for signal distortions, such as multiple signal peaks and range or Doppler spreading. The strength of the signals were also noted.

The sets of data with multiple peaks were time-averaged by mean averaging the multiple CITs and combining them into one. This time-averaging caused some of the noise to cancel out, thus reducing the noise level and making it easier to observe the signal. Data with multiple peaks or range or Doppler spreading were compared with data taken at the same time by a receiver from another experiment to verify the accuracy of the data.

Results

During the four-night span, the receiver collected a total of 93 sets of data with signals (Table 3-1). For all of those sets, the mean average signal-to-noise ratio was 25.7 dB and the highest signal-to-noise ratio obtained was 45 dB. Of all the data in which there was a signal, only about 21.5 percent produced a totally clean signal with one peak and little or no spreading (Figure 3-4). About 3 percent showed range spread and 24 percent had Doppler spread. The most important sets of data are those with multiple peaks. The distance between range peaks assist in determining the propagation modes of the signals. From the Doppler measurements, the relative velocities of parts of the ionosphere can be determined. As always, the possibility of instrumentation error must be considered.

Sweep Bandwidth	4.17 kHz	12.5 kHz	25.0 kHz
Number of Sets of Data with a Signal	35	53	5
Average Signal-To-Noise Ratio	32.4 dB	21.6 dB	22.0 dB
Largest Amplitude	45 dB	35 dB	30 dB
Percent with Totally Clean Signals	6%	26%	80%
Percent with Range Spread	8%	1%	0%
Percent with Doppler Spread	31%	21%	14%
Percent with Multiple Peaks	80%	11%	49%

Table 3-1. Table of some characteristics of the collected data.

Assuming the reflection of the signal by the ionosphere occurred at an altitude of 250 km during the times of data collection, there was a one-hop slant range of 1,350 km and 1,600 km for the two-hop propagation. Thus, any data with about 250 km between two range peaks had the possibility of having

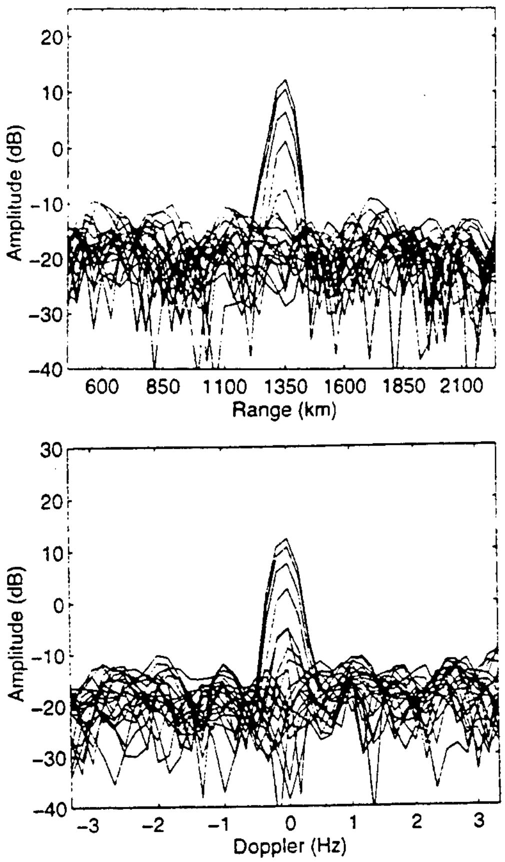


Figure 3-4. Example of a clean signal. The data of this signal was taken at 06:34 UT on January 13 using a sweep bandwidth of 4.17 kHz and a frequency of 6.211 MHz.

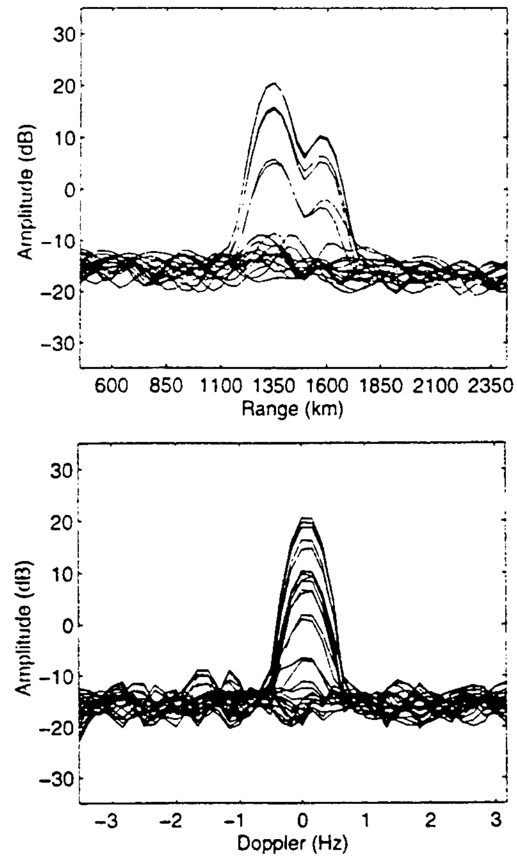


Figure 3-5. Range/Doppler graph of data with two range peaks. This data was taken at 05:26 UT on January 12 using a sweep bandwidth of 4.17 kHz and a frequency of 6.252 MHz. The four CITs of the data have been time-averaged. Note the lower level of noise compared to Figure 3-4.

undergone two-hop propagation. One such set of data was taken at 05:22 UT on January 12 using a frequency of 6.252 MHz and a bandwidth of 4.17 kHz (Figure 3-5). There are about 250 km between the two range peaks and, as would be expected, the second range peak is smaller than the first. The second, two-hop signal would be weaker because it would have hit the ionosphere at a greater angle than the one-hop, so more of the two-hop signal would have penetrated the ionosphere and caused additional propagation losses.

Another signal with two range peaks was received on January 12 at 05:33 UT (Figure 3-6). The transmitted signal had a frequency of 6.251 MHz and a 4.17 kHz bandwidth. Between the two range

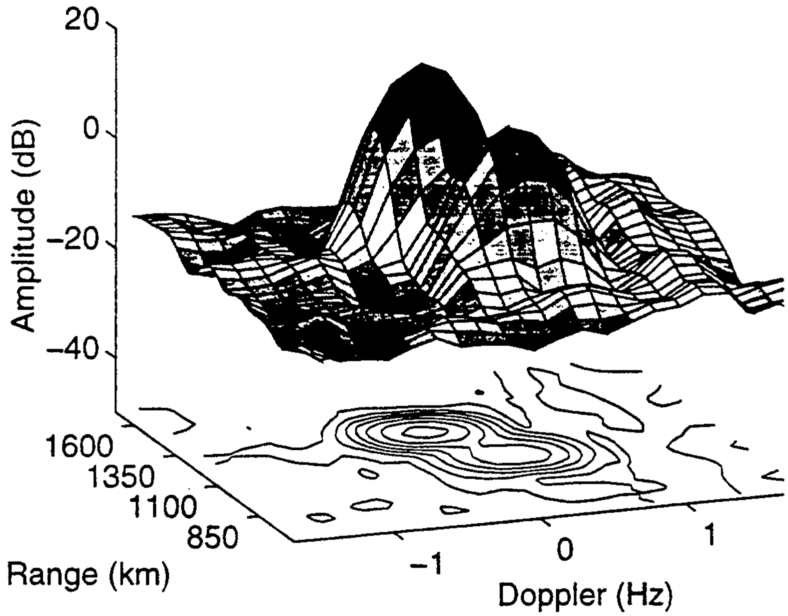


Figure 3-6. Combination surface plot and contour plot of data taken at 05:33 UT on January 12 using a sweep bandwidth of 4.17 kHz and a frequency of 6.251 MHz. The four CITs of the data have been time-averaged.

peaks are 215 km, making the data a good candidate for two-hop propagation. However, in this set of data, the second signal is the stronger of the two. If it did indeed follow a two-hop path, this suggests there was a strong ionospheric irregularity where the second signal reflected off of the ionosphere. Although such strong irregularities are less likely at mid-latitude, they are possible.

At 02:05 UT on January 13, a 9.200 MHz signal was transmitted with a 12.5 kHz sweep bandwidth. The received signal had two range peaks about 1,030 km apart. This large distance between peaks suggests they were caused by some sort of instrumental error or some unexplained phenomena. Some unusual, seemingly unexplainable results taken at high latitudes have been explained by the possibility of "ionospheric tilting" [K. B. Baker and R. A. Greenwald, 1988]. However, the mid-latitude ionosphere is more tame than it is at high latitudes, so there probably was no "ionospheric tilting" in any of this data.

Date	Time (UT)	Sweep Bandwidth	Frequency (MHz)	Distance Between Range Peaks (km)	Velocity Based on Difference Between Doppler Peaks (m/s)
January 12	04:41	25.0 kHz	6.234	138	150
January 12	05:22	4.17 kHz	6.252	251	0
January 12	05:33	4.17 kHz	6.251	215	16
January 13	02:05	12.5 kHz	9.200	1,032	213
January 13	05:01	4.17 kHz	6.237	107	24
January 14	01:15	12.5 kHz	6.240	72	0
January 14	05:53	4.17 kHz	6.245	179	0

Table 3-2. Table of data with multiple peaks.

Several sets of data taken on January 12, 13, and 14 had range peaks less than 250 km apart (Table 3-2). Such data could have been caused by either high ray propagation or off-axis scattering. During the experiment, the transmitter was supposedly operating close to the MUF all the time. Therefore the receiver should not have received signals caused by high ray. However, a great enough change in the strength of the ionosphere would have supported the high ray propagation. The existence of large signals due to off-axis scattering is questionable. Irregularities with very high electron densities could scatter signals off-axis with relatively high signal strength, but such irregularities are unlikely at mid-latitudes.

Data for the backscatter ionograms taken while the experiment was conducted are not yet available. Once that information is available, it will be compared with the data from the receiver to support or disprove the preliminary results. These comparisons with the ionogram data will be reported in the future.

Conclusion

In general, data with about 250 km between range peaks were assumed to be two-hop. Those with second peaks more than 250 km away from the first peaks were presumed to have been off-axis scattered, especially when the second peaks were much smaller than the first peaks. Data with less than 250 km between peaks were assumed to be caused by either off-axis scattering or high ray propagation. It

is still uncertain whether off-axis scattering is a significant contributor in degrading the signal quality of HF signals. These results will be more conclusive once the data from the backscatter ionograms are available.

References

Aarons, J. and S. Basu, "Scintillation on trans-ionospheric radio signals," in *Handbook of Geophysics and the Space Environment*, A. S. Jursa, ed., Hanscom AFB, MA: Air Force Geophysics Laboratory, 1985, pp. 10-71 to 10-83.

Baker, K. B. and R. A. Greenwald, "The vertical angle of arrival of high-frequency signals propagating from Thule to Goose Bay," *Johns Hopkins APL Technical Digest*, 9(2): 121 to 130, 1988.

Franchi, P. R., and E. J. Tichovolsky, "Phase screen modulation as a source of clutter related noise in over-the-horizon radars." Rome Laboratory, Tech Report No. RADC-TR-89-296. 1989.

McNamara, L. F., "Morphology of the ionosphere," in *Handbook of Geophysics and the Space Environment*, A. S. Jursa, ed., Hanscom AFB, MA: Air Force Geophysics Laboratory, 1985, pp. 10-47 to 10-51.

Millman, G. H., "HF radar ionospheric clutter," in *Handbook of Geophysics and the Space Environment*, A. S. Jursa, ed., Hanscom AFB, MA: Air Force Geophysics Laboratory, 1985, pp. 10-63 to 10-71.

Acknowledgments

The data collection was conducted by members of Rome Laboratory, including Timothy J. Tanigawa, Neil B. Myers, David S. Choi, Bertus Weijers, and Stuart L. Matthews.

Thanks to Timothy J. Tanigawa, Bertus Weijers, Neil B. Myers, David S. Choi, Peter Franchi, Stuart L. Matthews, and Belinda S. Chen for their help in writing this paper.

A STUDY OF THE RADAR SYSTEM AND THE C PROGRAMMING LANGUAGE

James R. Decker

Sauquoit High School
2601 Oneida Street
Sauquoit, NY 13456

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base,DC

and

Rome Laboratory

August 1995

A STUDY OF THE RADAR SYSTEM AND
THE C PROGRAMMING LANGUAGE

James R. Decker
Sauquoit High School

Abstract

In this research experiment the radar system and the C programming language was studied. The C programming language was learned to aid in the placement of the radar systems into an existing virtual reality environment. The graphics language of C was to assist in placing radar systems into the virtual reality environment.

A STUDY OF THE RADAR SYSTEM AND THE C PROGRAMMING LANGUAGE

James R. Decker

Introduction

In this research experiment the radar system and the C programming language was studied. The C programing language was learned to aid in the placement of the radar systems into a virtual reality environment. The graphics language of C was to assist in placing radar systems into the virtual reality environment.

Methodology

The C programing language is a very powerful and flexible programming language to use. Here are some of the different commands and statements that were learned. A variable is a named storage location in your computer's memory. By calling the named variable, you are calling the data stored in the variable. A constant is very similar to a variable. The only difference is a constant value can not be changed unlike a variable. There are two different types of constants, literal constant and symbolic constant. A literal constant is a value that has to be typed into the program. While on the other hand, a symbolic constant is represented by a name declared in the source code. Actually all you have to do is type in the declared variable's name and it's value will be entered into the program. There are seven different types of

numeric variables. This is because numeric values need different amounts of storage space. In using the correct variable, the program will run more efficiently. An integer variable will hold the value of whole numbers only. When an numeric variable is initialize, it instructs the compiler to set aside storage space for the variable. Initializing a variable is a good idea so that you will know what the value of the variable is.

A statement instructs the computer to execute a certain task. They always end with a semicolon and only one statement per line. An expression is anything that evaluates to a numeric value. Complex expressions are just simpler expressions that are connected by operators. Operators are symbols that tells C to carry out some operation or action on any number of operands. Operands are something that an operator acts on, in C, all operands are expressions.

Math operations such as addition and subtraction are performed by mathematical operators. There are several different mathematical operators. The first one which is addition (+) will add the two operands. The second, subtraction (-), will subtract the two operands. Multiplication (*) will multiply the two operands. Division (/) divides the first operand by the second operand. Modulus (%) will give the remainder when the two operands are

divided. Increment (++) will increase the operand by one and decrement (--) will decrease the operand by one.

Relational operators are used to see if there is a comparison between two expressions. The if statement will execute a command or group of commands only if it meets the value of the statement. A logical operator will allow you to combine two or more relational expressions into a single expression that evaluates as either true or false. By using parentheses the computer will put a higher precedence on an operation. A compound assignment operator is a shorter method for combining a binary mathematical operation with an assignment operation.

A function is a separate section of the C source code which performs a certain task and optionally returning a value to the program. Once the function is defined in the program; it then can be used anywhere in the program by simply typing the functions name. When the function is called it can be sent one or more arguments. An argument is program data needed by the function to perform its task. The statements are then executed performing whatever task it is designed to do. Recursion is where a function calls itself either directly or indirectly.

Arrays are an indexed group of data storage locations that have the same name and are distinguished from each other by a

subscript, or index - a number following the variable name, enclosed in brackets. The for statement will execute a block of one or more statements a certain number of times. The while statement is much like the if statement. It will execute the statement or block of statements as long as the condition is true. When a loop is nested; it means that there is a loop within a loop.

The printf function can do many different procedures. Mainly it is used for displaying text on to the screen. The printf statement can also display the value of variables. Also an escape sequence can be added into the statement which can ring a bell, start a new line, or any number of different sequences.

Pointers are one of the more difficult sections to learn in C programming. Pointers deal with the storage of data and where the data will be stored in the memory of the computer.

The graphics language of C was also utilized so the radar system could be placed into the virtual reality environment. The lineto command is the basic graphic command. It does what it says it draws a line to the point declared. A number of different shapes can be made with this command. The rectangle command draws a rectangle without using as much space in the source code as would the line command. The floodfill command fills in the region that is inside the

point declared. The ellipse command draws an ellipse with the coordinates that are given. With the ellipse command a sphere can be drawn by drawing a number of different ellipse.

The radar system is a very complex and difficult subject to study. The radar system was studied so the range of the radar could be calculated by using the radar range equation. There are many different factors that can affect the range of the radar. These include elevation and azimuth beamwidth, range, elevation and lobe boresight to name a few. This is why calculating the range of the radar was so difficult.

Conclusion

Although I did not finish as much of the project as I would have liked, I have still learned many things. I never had the time to put the radar system into a virtual reality environment. Learning about the C programming language and about radar systems will be very helpful in the future.

References

Aitken, Peter, and Bradly Jones Teach Yourself C in 21 days. Indianapolis: Sams Publishing, 1994.

Skolnik, Merrill I. Radar Handbook. New York: McGraw-Hill Book Company, 1970.

DATABASE CREATION
USING EXISTING SOFTWARE

Mary X. Fitzgerald
High School Apprentice/IRDS

Holland Patent High School
Thompson Road
Holland Patent, NY 13435

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

DATABASE CREATION USING EXISTING SOFTWARE

Mary X. Fitzgerald
High School Apprentice/IRDS
Holland Patent High School

Abstract

The purpose of this research was to develop and populate procurement and timeline databases. To develop the databases, the researcher had to become familiar with the database application program and the type of information to be contained in the database. The first database created was the Procurement Database for the engineers in IRDS. The Procurement Database simplifies and automates the paperwork procedure for laboratory acquisition programs. The application used to create this database was FileMaker Pro version 2.1. The second database created used the Timeline Analysis System (TAS) as the program. This database's purpose was to place medical data on TAS since it had not been tested for this use. Netscape Navigator version 1.1 was used as a browser for research on the World Wide Web. Unfortunately, there was not sufficient medical data to complete a medical database . There was information on the Web but there was not a sufficient amount to test TAS's ability to use medical data. Instead TAS was tested on its ability to use Historical data. Information was gathered from the local library and The Rome Historical Society. Once the preliminary research had been accomplished, work began on the actual databases.

DATABASE CREATION
USING EXISTING SOFTWARE

Mary X. Fitzgerald

Introduction

For each contractual package, IRDS completes several form types repeatedly. Every time there is a change in the package, the forms must be redone. Often the changes are as small as a new number. There is common information found on most of the forms. This project was to combine the information on the forms into one database that would populate all forms accordingly. The software package FileMaker Pro was used for this purpose. FileMaker Pro allows the user to have different layouts to support the different acquisition forms.

Methodology

Most of the forms to be included in the database were imported from preexisting files in other software programs to an empty file. The imported forms became the layouts in the Procurement Database file. The Data Input Fields and the Quad Chart layouts were created in FileMaker Pro. The forms were then populated with fields based on the type of information each contained. Information found on two or more forms was created as one common field. The fields were created as text fields. Later some of the text fields were changed to calculation or summary fields to make the database simpler to use. The fields were originally formatted as single entry fields. As with the field type, a few changed to repeating fields to simplify use of the database.

At the beginning of the project a FileMaker Pro database existed for the RL 2913 Status Report. A majority of the current IRDS projects had been inputted in that file. The records from the RL 2913 database were imported to the Procurement Database. The information from the RL 2916 Program Implementation Request and Authorization was matched with the records already present in the database. If the project data had not been entered, it was then entered.

The Data Input Fields were then created. The Data Input Fields is the one layout that contains all the information on all the forms. The common fields, that were on more then one form, were placed in a Common Area at the beginning of the Data Input Fields. The rest of the layout was grouped by forms. All information found on one form was placed in one area.

With the Data Input Fields beginning to take shape, scripts and buttons became useful to have. Scripts to print each individual layout were written. These scripts were then placed as buttons at the top of the Data Input Fields. Scripts were also added to the top of the Data Input Fields to enter the Find mode, Sort Records, Create a New Record, Move Back and Forth between the records and an all-purpose Print. A special script was written for the importation of the picture to the Quad Chart and to make sure the page set-up would be correct for printing. Another special

script was written for the Cost/Analysis chart and program schedule on the RL 2913. It was placed as a button within the area designated for the RL 2913.

An engineer used the database for a new contractual package and offered some observations. One observation was that some of the fonts used were not common fonts and more common fonts were substituted for the illegible ones. There was also confusion in areas due to the single entry fields. Those problem areas were changed to repeating fields. Some fields involving finances were changed from being a text field to calculation fields. As calculation fields, certain field values would not have to be computed by the engineer. If one of the numbers changed, the database would reflect the change throughout the calculation fields. With the change to calculation fields, more fields could be placed under the Common Area.

Conclusion

The largest problem encountered was that some of the imported documents were not correct. Some of these problems were correctable, a few were not. The AFSC 2857, AFSC 2944 and the Quad Chart were not placed on the Data Input Fields. To reach these documents, there are buttons. These documents were left of the main form because of there layout. The database contains eight forms and forty records and is available to any of the engineers within IRDS. A User's manual was written for the Procurement Database. Most of the engineers also received a brief training on how to use the database. An engineer can now sit down complete one form and all the forms for the contractual package will be populated. This task has been completed successfully with the exception of incorrect imported layouts.

Introduction

An engineer within IRDS created an application program known as Timeline Analysis System or (TAS). TAS was originally developed for military use. The program places icons on a timeline. TAS allows the user to visualize and analyze past events displayed as icons. TAS currently allows its users to chose from approximately fifty-five icon types in a variety of colors. After the icon is chosen, the event data and any text may be added. The program allows the user to set the timeline parameters and number of days viewed.

The task was originally to apply TAS to medical events. However, access to sufficient medical data was not available. Events from history were used instead of the medical data. Historical Events from Rome, New York during 1750 to 1800 and 1950 to 1982 were entered. Historical events of the world during 1950 to 1954 were also inputted. The amount of data available and the time frame were the basis for the creation of several different timeline formats. In total, there are six timelines formats. The first two were based on local events, the other four on world history.

Methodology

The first timeline entered was from the 1950 to 1982. The data was found in a pamphlet from the Rome Historical Society. The data was already formatted as a timeline and seemed a logical place to begin. The timeline began on January 1st, 1950 and spanned the next thirty-five years. Events were placed on the exact date, if given but often only the year was given. When the year was the only part of the date given, the event was placed on January first of the year. Each of these events were given a designator of Rome. (A designator allows the user to chose a subset of events, rather then the completed database.) This subset of events was also stored as a classic entitled History of Rome 1950. The name was later changed. (A classic saves the timeline format, background color and the event location.)

The second timeline dated from 1750 until 1800. For this timeline, information was gathered from the pamphlet from the Rome Historical Society and a booklet about the Battle of Oriskany. Each of the events on this timeline, has a designator called the Battle of Oriskany. The designator denotes that this subset of events took place in the eighteenth century. In addition to the Battle of Oriskany designator, some also received the designator of Rome. This subset of events pertains only to events that occurred in Rome, New York. The other events on this timeline were specific to the Battle of Oriskany fought during the Revolutionary War. The icons used in the first and second timeline are the same for similar events.

The remainder of the timeline formats came from a Historical Reference book. Each of these timelines span two years. The timeline formats are 1950-1951, 1951-1952, 1952-1953 and 1953-1954. The events were placed into eighteen rows. Each row is a category. Within the categories,

different icons and icon colors denote corollary events within that category. An example is that one of the rows is composed entirely of folders. A red folder deals with "The Red Scare" or McCarthyism. A orange folder deals with the US's foreign policy. This category deals with political activities of the US. Each row has a different but very broad perspective. On this set of timelines, there were so many dateless events that they were spaced out along the timeline for better viewing. As with the first two timelines events with out a specific month or date were entered as January first of that year. However if it was the second dateless event, it was placed on February first. If there were more then 12 dateless events, the thirteenth was placed on the fifteenth of January. Each additional event after that was placed in the next month on the fifteenth. With those exceptions, each event was placed on the date it occurred.

<u>Conclusion</u>

This task was completed successfully. A user can now view several historical timelines on TAS. Some of the icons might not be entirely appropriate symbols for the event. This is due to the fact that TAS was originally developed for a different purpose and currently there is no method to add new icons. It was also unfortunate that many events gave no specific date. With a lack of specific dates, a user loses some of the ability to analyze the events. The result of this effort is several informative timelines that give the user a visual aid to historical events and their relation to other historical events.

References

40 Miles of American Heroism: A guide to General Herkimer's historic line of march, August 3-6, 1777. Syracuse, NY: Marshall Penn York Co.,Inc., 1977.

Scott, John Albert. Rome, New York: *A Short History*. Rome, NY: Rome Historical Society, 1983.

Trager, James Editor. The People's Chronology: A Year-by-Year Record of Human Events from Prehistory to the Present. New York: Holt, Rinchard and Winston, 1979.

INVESTIGATION OF A NETWORK CONFIGURATION AND DEVELOPMENT OF A UNIX NETWORK APPLICATION

David Gurecki

Rome Catholic High School
800 Cypress St.
Rome, NY 13440

Final Report for:
High School Apprenticeship Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

INVESTIGATION OF A NETWORK CONFIGURATION
AND DEVELOPMENT OF A UNIX NETWORK APPLICATION

David Gurecki
Rome Catholic High School

Abstract

In this age of telecommunications, the ability of computers to transfer data between different systems becomes increasingly important. While networking technology is not new, it has progressed substantially in bandwidth capabilities. Similarly, the volume of data needed to be transferred has increased, as seen in the bandwidth demands of full-motion video teleconferencing. Consequently, the continuing research and development of network technology to utilize the new capabilities continues to be an important field. Given the complexity and size of networks in such research and development facilities, an accurate representation of the network nodes and connections is extremely useful. A map was created of the network configuration at Rome Lab's Network Design Facility (NDF) using a network mapping application. This involved the collection of data describing the network nodes and how they are connected over several Ethernet and Fiber-optic ATM networks. Upon completion of that task, the development of a network application was explored. A whiteboard drawing program was ultimately written to be used with a muticasting Asynchronous Transit Mode (ATM) network interface being developed. Writing the whiteboard program involved creating a drawing interface, and developing methods for exchanging data between multiple computers over a network. The program was completed and successfully tested in the multicasting environment.

INVESTIGATION OF A NETWORK CONFIGURATION
AND DEVELOPMENT OF A UNIX NETWORK APPLICATION

David Gurecki

Introduction

The worldwide communication of data has been revolutionized by the ability to network computers together. There are many issues and concepts involved in computer networking. How is networking accomplished? What speed can information be transferred? Can multiple computers be conferenced together for data transfer instead of point-to-point connections? These are some of the concepts encountered during this summer's research, and which bear discussing before delving into the research itself.

Networking is accomplished using one of two primary media: electrical wire or fiber-optic cable. An Ethernet network primarily uses wire to communicate electric signals between two computers. An ATM network generally uses fiber-optic cable to communicate light signals between two computers due to its higher capacity. Ethernet technology has existed for several years, while fiber-optic cable is comparatively new and less widely used. ATM networks, however, have a much higher bandwidth, i.e., can transfer data at a much higher rate. Computers on an Ethernet network share connections to a common wire that runs between the computers. In an ATM network, by contrast, each computer is connected directly to an ATM switch since fiber-optic cables generally communicate point-to-point. It is possible that a computer can be connected to multiple Ethernet networks and an ATM network.

The speed at which information is carried across a network medium is referred to as its bandwidth. There is a growing demand for higher network bandwidths, as seen in real-time applications such as real-time video and sound. The bandwidth of a typical Ethernet network is 10 megabits per second, and the bandwidth of a typical ATM network is 155 megabits per second. Since ATM networks are much more suited to handle that load, they will undoubtedly become more commonly used. At the present time, however, ATM is in its infancy and a number of technical issues need to be resolved before it becomes a usable media.

As ATM matures, there will be a demand for software that takes advantage of ATM technology for transferring information. One advantage of ATM technology will be the ability to multicast information. Traditionally,

sending information from a source to many remote systems requires that an individual connection must be established between each remote computer and the local source. In that case, the same information will be sent over the Ethernet cable multiple times (one copy for every remote system). This is inefficient and a waste of network bandwidth. In an ATM network, every computer is connected to an ATM switch. This allows the source computer to instruct the ATM switch to replicate the information and send it to several remote computers. The source then only has to transmit one copy of the information, and the switch will "multicast" it to several destinations. Therefore, not only is an ATM network inherently faster than Ethernet, it can also send information from one source to many destinations much more efficiently. Network applications that take full advantage of this capability will be essential for future communication demands.

Discussion of Problem

Two individual tasks were pursued during this summer's research period: mapping of a network configuration and the development of a UNIX network application.

In the NDF of Rome Laboratory, there is a very complex network involving many computers connected to both ATM and Ethernet networks. However, there was no current diagram of the network configuration showing how the computers were connected. The need was recognized to compile a database of information about each computer and develop a diagram that showed how each computer was connected to the different networks so as to provide a clear, easy to understand reference for use by people who use the network systems on a daily basis.

The second task involved learning how to develop a network application, ultimately to be used over an ATM multicasting network interface. This work was done as a collaborative effort with a group of graduate students from Kansas State University who were developing an ATM network interface that takes advantage of ATM's multicasting capabilities, described earlier. This effort was to develop a program to send live video, sound, and other media information from a single source to multiple destinations. As part of this experiment, they wanted to test a whiteboard drawing program with their network interface. The development of such a program became my project. This whiteboard application would allow the local user to draw a diagram using the mouse as a "pen". The points drawn on the local screen would then be transmitted to destination windows, which would draw the points in the same location on

their screens. All the screens would maintain an identical picture. Additionally, it was a design requirement that any of the drawing programs would simultaneously accept user input and receive input from other drawing windows. This is in contrast to having a single input source and multiple "viewing-only" windows, as is the case with live video transmission. Also, the user should be able to change pen and background color, and the width of the pen.

The standard interface for writing graphical user programs for the X Windows system is Motif. It provides many devices for accepting input from and displaying output to the user. Pushbuttons are one such input device. Pushbuttons can be programmed to execute a certain set of commands when activated by the user. For example, in this program pushbuttons are used for changing color, pen width, or closing a connection. Motif's drawing area is another device used by this program. "Drawing Area" is a misnomer, however, because Motif does not initially recognize mouse input or draw any lines as the name might suggest. The programmer is responsible for handling mouse input and creating and maintaining the image in the window using lower-level graphics routines.

Although the ultimate goal was to develop a multicasting application, a server/client interface was initially developed for simplicity. In this model, only two copies of the program communicate with each other at one time. This permitted a method of packetizing the data and other interfaces to be developed first without the concern of the multicasting interface. This initial test system was written using both Ethernet and ATM library routines. Once the initial system was working successfully, it was adapted for the multicasting software.

Methodology

Network Configuration Mapping

The first project involved the creation of a diagram to represent the network in the NDF at Rome Lab. First, the software package NetViz[1] (version 1.2b), was investigated. This package allows the creation of multiple-level diagrams composed of nodes (representing computers/gateways) connected together with links (representing Ethernet or fiber-optic lines). The user must create a catalog of nodes and links which are to be used in the diagram before constructing the diagram. The catalog defines the names of all the nodes, which graphic to use for each node, the names

[1] NetViz is © Copyright 1994 by Quyen Systems, Inc.

of its attributes, and which attributes are displayed with the icon. It also defines the names, styles, and attributes of the links.

Next, the network information had to be gathered to create the catalog. An inventory of the names of all the computers, their IP addresses, types, and operating systems was compiled. The information regarding the configuration of all connecting Ethernet and fiber-optic interfaces was obtained. Finally, the node information was entered into a database, and a catalog was generated using all of this information.

At this point, enough information had been collected to begin constructing the diagrams. The NDF network is divided into 4 sub-networks. A top-level diagram, which consists of an icon for each sub-network, was created (See Figure 1). A blank network-map page was created for each sub-network and linked to its respective icon in the top-level diagram.

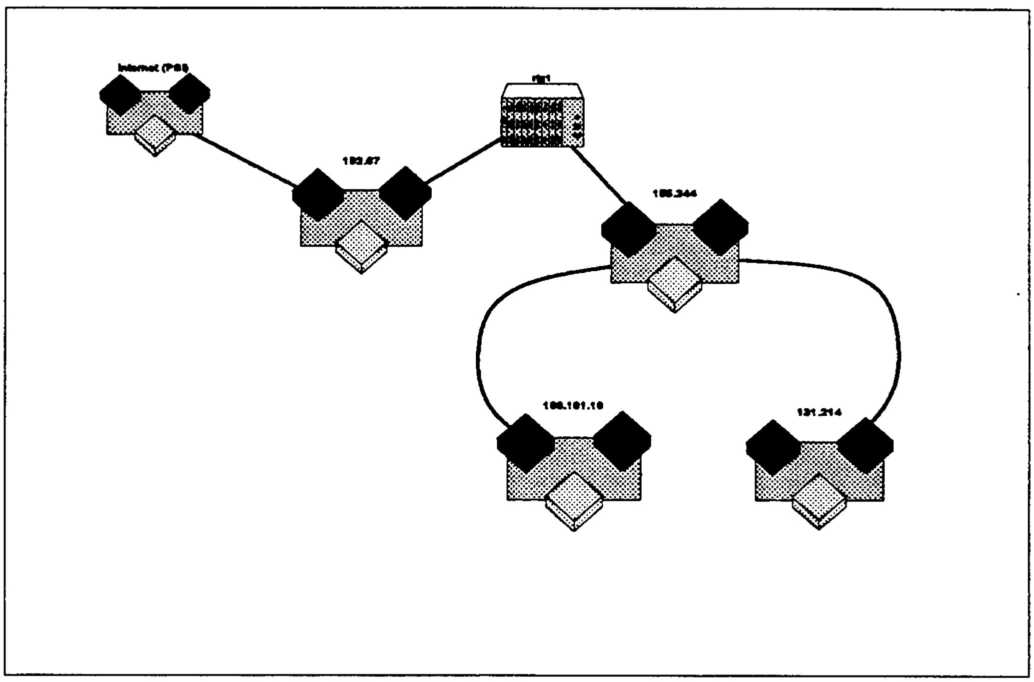


Figure 1: Top-level network map

The database of information collected earlier was then imported directly into the blank map pages. Using the "Import Data" function of NetViz, icons for each computer in the network were automatically generated. NetViz also imported the various attributes for each node in the project. (That information can be viewed by selecting a node on the screen and activating the Inspector.) At that point, the icons were arranged into their appropriate sub-network, and

linked with the appropriate network connections (See Figure 2 for an example of a sub-network map).The only attribute that was not stored in the nodes was the IP address of each computer. Instead, the address was shown on the link that connected each node to the network. This was because some computers have more than one IP address even though they are a part of the same network. It is also easiest to represent the multiple IP addresses on multiple Ethernet lines. Furthermore, it is conventional to show the IP address in association with the connection to the computer, not with the computer itself.

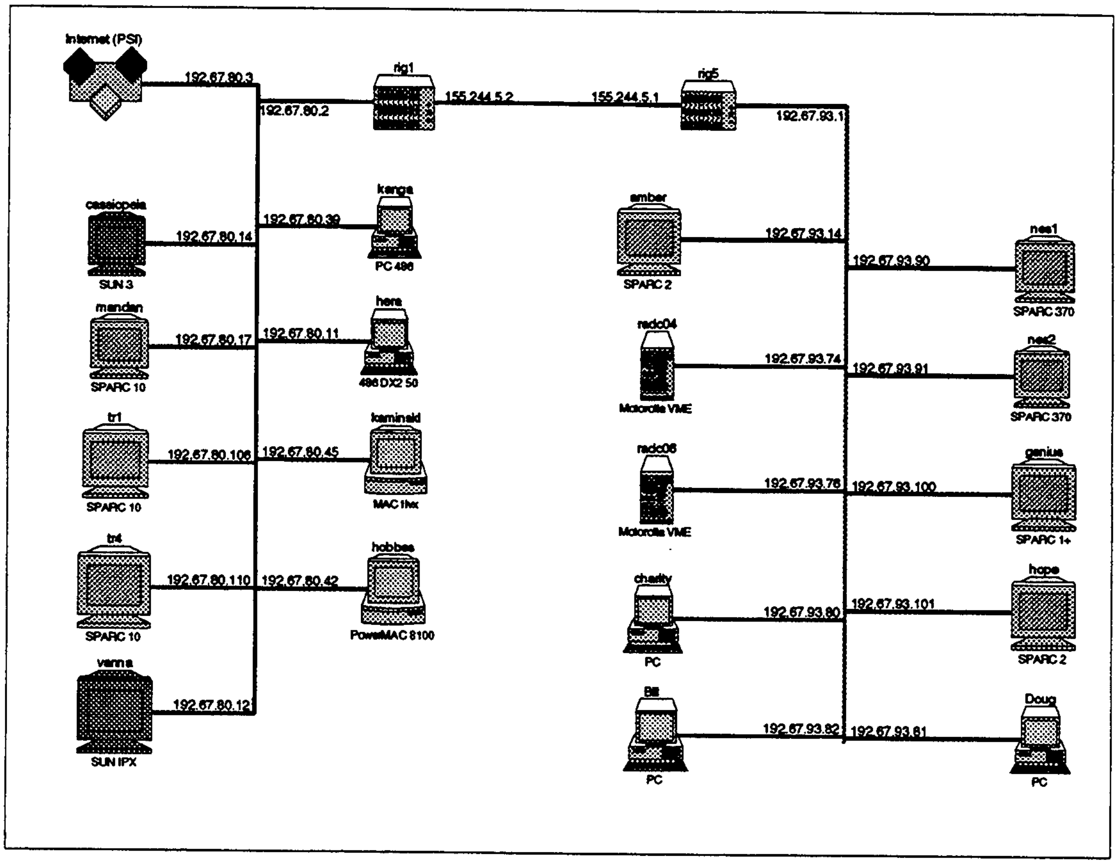


Figure 2: One of the sub-network maps

Whiteboard Program Development

The first step in the whiteboard development was to learn how to program in Motif, which is a set of widgets, or objects which provide an interaction between the user and the program. Examples of widgets include buttons, scroll-

bars, menus, a drawing area, and many other devices. Each widget has a set of resources, or characteristics. For example, a push-button has a resource which determines the size of the shadow. Some widgets have callback resources which are set by the program to link routines to an action taken on a widget. For example, a set of code can be linked to execute every time a button is pressed.

Once the basic concepts had been studied, development began on a simple drawing program, initially with no network connectivity. The window layout was designed to have a row of pushbuttons along the top (containing the "Close", "Pen Color", "Background Color", and "Pen Width" buttons), a 400x400 pixel drawing area beneath it, and a frame around the drawing area (See Figure 3).

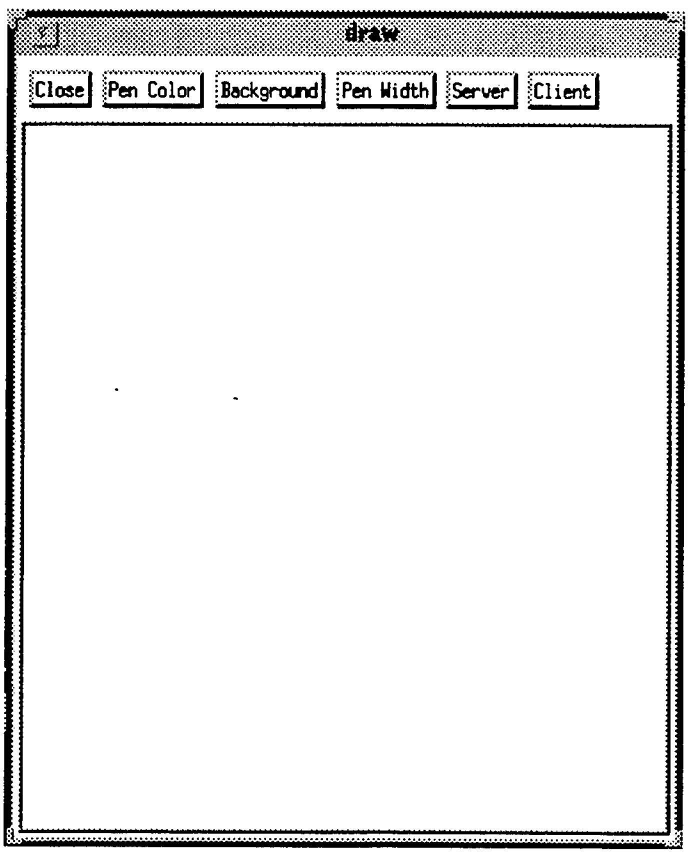


Figure 3: Main Window

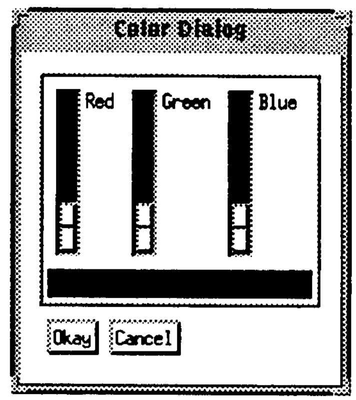


Figure 4: Color Dialog

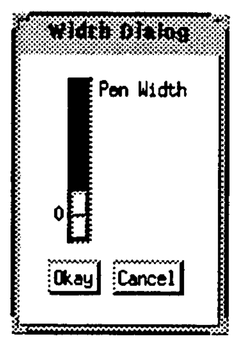


Figure 5: Pen Width Dialog

When the code to display this window layout had been completed, the drawing area input and output routines had to be written. The goal was to allow the user to draw by pressing the left mouse button down, moving the mouse, and releasing the mouse button to stop drawing. To incorporate these actions into the program, routines had to be written to recognize input from the mouse (button presses and motion), draw lines under location of the mouse pointer,

and store those locations in memory.

First, the memory routines were written to maintain an array of drawn points in memory. Next, the previously mentioned mouse actions were associated with a drawing routine by specifying an "Action Table" with the drawing area. Code was then written to add a point to the array and draw it on the screen when the mouse button is pressed and the mouse moves. Also, the right mouse button was associated with a routine to clear the window, and the middle mouse button was associated with a routine to remove the last pen stroke off the array ("undo"-ing the last pen stroke). Finally, a routine was written to redraw all the points stored in memory whenever the drawing area is re-exposed, such as when the window is de-iconified or when other windows are placed over the drawing window and then removed.

In order to change the pen color, a color dialog box had to be created. A dialog box is a stand-alone window that interacts with the user. A color dialog box was created to allow the user to change the red, green, and blue components of the pen color by moving sliders along three scales (See Figure 4). Whenever one of the scales is moved, the new color is displayed in a color bar, located just below the sliders. When the user has selected the desired color, he presses an "Okay" button to update the pen color in the drawing window. If the user selects "Cancel", the pen color will be unchanged. The new color does not affect lines already in the window, only lines drawn subsequent to the color change. Finally, the color dialog box was configured to activate when the "Pen Color" or "Background" button on the button bar is pressed.

In a similar manner, a pen width dialog box was created. It contained a single slider that allows the user to change the width of the pen (See Figure 5). Again, selecting "Okay" updates the pen drawing width and selecting "Cancel" closes the window without updating.

At this point, development began on the exchange of data over the network. Initially, the program was designed to communicate with one remote site over an Ethernet network. First, a method was designed to handle the transfer of information between computers. The information would be placed into a "packet" which contains a command and four bytes of special data. The command refers to what action to take (draw a point, change the color, clear the screen, etc.), and the data refers to any specific data associated with the command (the coordinates of the point, the new color, etc.). Therefore, every time the mouse is moved with the button down, a point will be drawn on the screen, added into memory, and sent over the network in a single packet. The receiving side would be constantly

monitoring the Ethernet port for incoming packets, and would process any that come in.

Next, the UNIX routines available for network communication were studied and implemented into the program. In order to establish a connection between two computers, one must be designated the "server" and the other the "client". The server must be waiting for the connection to be made, and the client initiates the connection. Once the server and client have established a connection, both sides can send and receive information. Hence, both computers could accept input from their user as well as the remote computer. Two buttons were added to the button bar to enable the user to initiate their computer as either the server or the client (See Figure 3).

After this was tested over an Ethernet network, the code was adapted to operate over an ATM network using FORE[2] Systems' C library routines. This was uncomplicated since the ATM routines follow the same server/client model as the Ethernet routines.

Subsequently, the program was adapted for use with the Kansas State research group's multicasting interface. Their program consists of a transmitter and a receiver program, plus additional applications on each computer. The components communicate with each other via UNIX shared memory. For example, if a video frame is received by the receiver program, it is placed in shared memory and the video viewer application is notified. That application then removes the frame from shared memory and displays it on the screen. When a local video window captures a frame, it places it in shared memory and notifies the transmitter. The transmitter then multicasts the data to the multiple recipients and removes it from shared memory. Since the whiteboard drawing program was to become an end-user application of the multicasting software, it had to be modified to use the shared memory and communicate with the transmitter and receiver components. The network initialization routines were removed and replaced by the research group's routines for initiating a multicast connection. Also, the whiteboard's transmit and receive routines were modified to use the shared memory for sending and receiving data, instead of sending it directly through the network as in the server/client models. The Server and Client buttons were removed from the button bar, since all the connection information is set up before the window appears.

This drawing program is unique from the video and sound applications. For video multicasting, a single machine needs to be transmitting frames taken from a camera, and remote machines need to have a viewing window

---

[2] FORE Systems, the manufacturer of the ATM switch, provides programming libraries for setting up ATM connections.

open to receive the information. Each viewing window is only *receiving* from the network, and the source computer is only *sending* to the network. The whiteboard requires that each computer accept local drawing input as well as input from any of the other computers, i.e., it needs to simultaneously send and receive information. If the user draws on a local computer, every remote computer in the conference must reflect the drawn line on its window.

A method was devised to enable every copy of the program to transmit to every remote system and receive from every remote system. Sending to every other computer requires only that a single packet be sent out, since the ATM switch multicasts the data to every remote computer. However, the receiving connections to every other computer must constantly be monitored for incoming data. This requires that several tasks be running concurrently to handle the sending of commands to and the receiving of commands from the remote computers. This was accomplished using UNIX's ability to "fork" child processes from a single parent process. A separate process is spawned for the sending

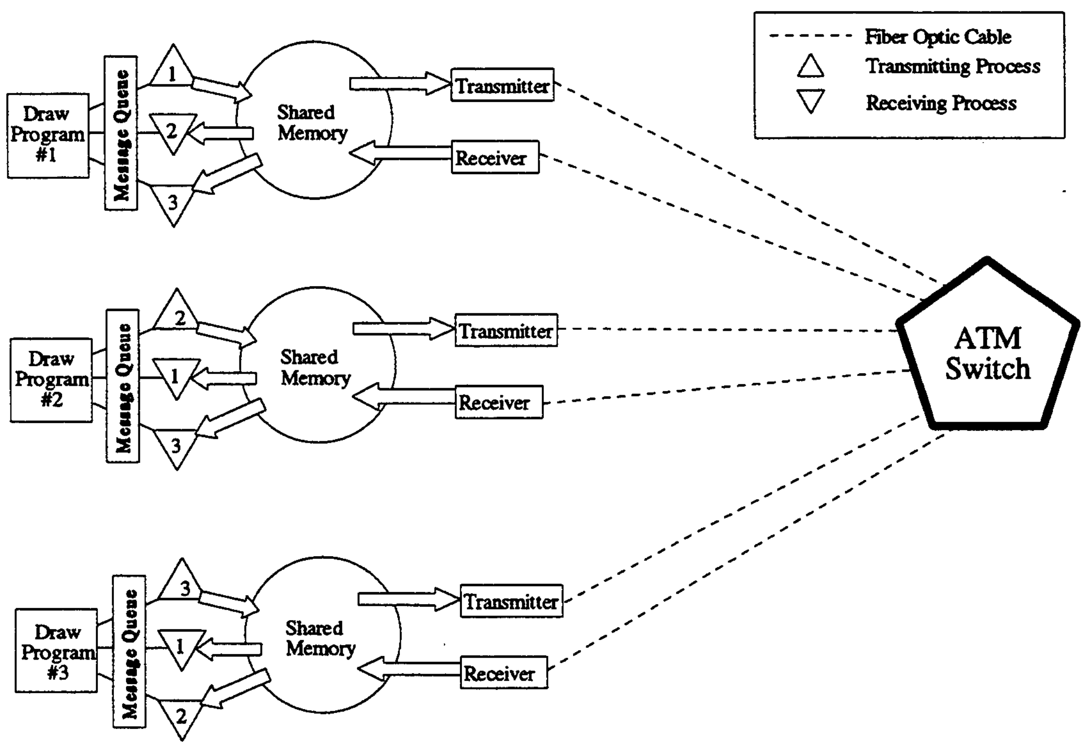


Figure 6: Draw Multicasting Model

unit and for each receiving unit (represented by triangles in Figure 6). These processes act as an intermediary between the draw program itself and the shared memory. The separate processes communicate data to the draw program by means of a UNIX message queue. See Figure 6 for a graphical representation of this model.

Consider a possible scenario with the preceding network model. If the user at draw program #1 draws a point, the draw program will create a packet with the "point" command and the coordinates of the point in the data buffer. It will place this packet in the message queue with a unique ID. The transmitting process (triangle labeled "1"), which is scanning the message queue for the same ID, will find the packet and place it into shared memory. The Multicast Transmitter will then take the packet and send it to the switch with the instructions to multicast it to draw programs #2 and #3. The packet will then be received by draw program #2's Multicast Receiver, which will place the packet in shared memory. Then, the draw receiver process which looks for packets from program #1 (upside-down triangle labeled "1") will find the packet, remove it from shared memory, and place it into a message queue with a unique idea. When the draw program next scans the message queue, it will detect the packet and draw the corresponding point on the screen. The same process happens with draw program #3.

Since the size of the packets being transmitted is so small (about 6 bytes), there is a great deal of overhead created every time a packet is sent. This is because at every step of the transmission process, more header information is added to the packet (involving size, sequence number, etc.). By the time the packet is traveling over the fiber-optic cable, the header gets to be as big, if not bigger, than the packet itself. To reduce the overhead, the fixed-length packet size was changed to a variable-length packet size. Instead of sending a packet every time a point is drawn, points are accumulated into the packet data buffer as they are drawn. When 128 bytes have accumulated, the packet is transmitted. If the mouse button is lifted before 128 bytes have accumulated, the packet is sent anyway. This way, the majority of the data being transmitted is actual point information, not header data. The tradeoff is that the remote terminals are updated periodically, instead of constantly, causing a delay between when points are drawn and when they are viewed on the remote terminals.

Finally, a chatting capability was added to the program. When the user presses the "Chat" button on the button bar, a chat dialog box window appears on the user's local screen and each remote user's screen. The dialog box consists of a single-line text box for inputting text, a "Send" button to transmit the text that has been typed, and a 5-line scrollable text window that displays the messages as they are sent and received. To chat, the user simply types a

message in the text-input line, and either presses Enter or clicks the "Send" button. The message will be preceded by a number corresponding to the entity address of the user's program (#1, #2, #3, etc. as shown in Figure 4). The message will then appear in the user's local text-viewing window and all remote user's text-viewing windows. The chat capability was made possible by the change to a variable-length packet model.

Results

Network Configuration Mapping

The map of the Rome Lab NDF network was completed successfully. The final version consists of one top-level map and four sub-network maps. The top-level map consists of icons representing the sub-networks. When the user double-clicks an icon, NetViz brings up the respective sub-network map.

Whiteboard Program

The whiteboard program "draw" was also successfully completed. It was tested in a multicast session with three computers connected together. Each of the three computers could both draw to and receive from the other two. The program was included in the Kansas State research group's demonstration of their ATM interface.

The only major problem that remains in the program is that it is unable to recover from packet loss. When multiple applications are all interfacing with the ATM multicasting program (especially live video viewers/transmitters), the system resources are quickly used up. Since the system is constantly occupied, it is possible that a packet may get lost and not reach the draw program. In the original server/client version of the program, an error-correcting mechanism had been implemented. If one computer lost a packet, it sends a request to the other side to have it retransmitted. However, once the program was moved to the multicasting model, this mechanism would not work. If one computer asks to resend a packet, all the remote programs would get the request and resend the packet. Every computer would receive multiple copies of packets it did not need corrected. In order for the mechanism to work, it would have to be modified to send the request to a specific computer. The resending computer would have to specify a destination, so that all the computers but the intended one would ignore the packet. While this modification is possible to implement, it was not determined to be a priority since packet loss was infrequent and did not always effect the drawing. For example, if a

user is drawing a line, and one point in the middle is lost, the program still connects the previous and next points together. If the packet lost contains another command, such as a color change or a clear screen command, it would be a serious problem, but it was found to be uncommon.

Conclusion

It is important to create and maintain an easy-to-use and accurate diagram of a network configuration. Furthermore, it is beneficial to be able to retrieve the information and store it into a database format, as NetViz allows, so that it may be easily imported into other system management software that may emerge in the future.

As old Ethernet networks are gradually replaced by fiber-optic networks across the country, applications such as live video teleconferencing will become more prevalent. An ATM interface that allows multicasting, such as the one the Kansas State research group is developing, will become invaluable for taking full advantage of the capabilities of the ATM environment. Video teleconferencing packages seldom include only the transmission of video and sound, and often include many other features, such as graphics sharing capabilities (e.g., a whiteboard), file copying, and sharing of database or spreadsheet application. The development of a whiteboard application that functions in a multicasting environment is a timely and worthwhile exercise.

ASSEMBLY LANGUAGE AND C DESIGN VERIFICATION AND TESTING OF A SCALABLE, PROGRAMMABLE, DIGITAL SIGNAL PROCESSOR

Eric J. Hayduk

Rome Catholic High School
800 Cypress St.
Rome, Ny 13440

Final Report for:
High School Apprenticeship Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington DC

and

Rome Laboratory / OCSS

August 1995

ASSEMBLY LANGUAGE AND C DESIGN VERIFICATION AND TESTING OF A SCALABLE, PROGRAMMABLE, DIGITAL SIGNAL PROCESSOR

Eric J. Hayduk
Rome Catholic High School

Abstract

Test programs that will be used to verify the design of the Floating Point Application Specific processor (FPASP5), a scalable, programmable, digital signal processor, were developed. The test programs were written using an assembly language specific to the FPASP5. Several programs written in C, a multipurpose programming language, were used to generate data tables for the Arithmetic Logic Unit (ALU) test program. Test programs were also developed to examine all registers, the incrementers, the C and D pointers, the C and D incrementers, all branch conditions that are available to an assembly language programmer and other assembly language commands. In addition, a flexible program was written to allow an individual test or multiple tests to be executed. Test programs and their usefulness in debugging the FPASP5 are discussed along with problems, corrections, and future steps to complete design verification.

ASSEMBLY LANGUAGE AND C DESIGN VERIFICATION AND TESTING OF A SCALABLE, PROGRAMMABLE, DIGITAL SIGNAL PROCESSOR

Eric J. Hayduk
Rome Catholic High School

Introduction

In recent years, a need has grown for computer processors that have the ability to handle floating point calculations quickly and efficiently. At the same time, it is essential that these processors have small size, low weight, and low power consumption in order to be useful in advanced signal processing applications. The Wafer Scale Signal Processor (WSSP), currently being developed by Air Force Rome Laboratory, is an adaptable and highly efficient processor that meets these requirements.

The fields of signal processing, which involves the detection and tracking of both ground and airborne targets, and supercomputing will benefit greatly from innovations in technology that allow the WSSP to place today's supercomputer performance on a single 6" x 6" board. Chip stacking, hybrid wafer scale integration, and a streamlined architecture design give the WSSP significant advantages over other signal processors. Chip stacking involves thinning down the backs of several chips and stacking them on top of one another to save space. Hybrid wafer scale integration is the process of producing chips and then placing them on a wafer, edge to edge, with minimal spacing. Finally, a streamlined architecture and improvements in memory packaging allow for a greater reduction in the size of the WSSP and better memory organization.

The unique FPASP5 architecture and a highly optimized software environment allow the WSSP to handle floating point operations more quickly and efficiently than other signal processors. Both the FPASP5 assembly language and microcode incorporates RISC-like instructions including loads, stores, and simple arithmetic operations. RISC (Reduced Instruction Set Computer) architecture utilizes simpler instructions to increase performance. This occurs because compilers, which optimize programming code, almost always use simpler assembly language instructions rather than more complex assembly language instructions. The FPASP5 software environment also incorporates critical high performance vector routines that can be called by high level language programmers. These routines are an integral reason for the increased performance of this signal processor.

Methodology and Procedures

In order to design the WSSP, several software models of the FPASP5 were created. (The WSSP is composed of FPASP5s and memory elements that take advantage of chip stacking technology.) The first model of the FPASP5 was created using Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL). This model was used mainly to debug the FPASP5 assembly language by running actual code on the simulated chip. The second software model of the FPASP5 was created using IRSIM, an event-driven logic-level simulator. In other words, IRSIM models each transistor and the circuit pathways between them. An output file created by running an assembly language program on the VHDL model of the FPASP5 is used as the event that stimulates the logic-level model. The logic-level model allowed our research team to debug and optimize the processor without fabricating a new chip every time an error was corrected or an optimization was made. In addition, the software models allowed our research team to debug, and optimize the software environment in parallel with hardware development.

A great deal of the design verification process has already been completed. The VHDL model of the FPASP5, the the FPASP5 microcode, function libraries that can be called by high level language programmers, and an assembler, which has the ability to convert assembly language programs into machine language, functioned accurately and efficiently. With minor exceptions, the FPASP5 assembly language also functioned accurately.

When writing any sort of program, however, one must remember that 40% to 60% of the time will be spent debugging. Currently, the WSSP project is in the final stages of the debugging process. Once the hardware design is optimized, the test code that is designed to check to functionality of the FPASP5 will be executed on the logic-level model. If any errors are discovered they will corrected and the model will be reevaluated. After an accurate logic-level model is developed, the WSSP chip will be fabricated. The complete battery of test programs will then be run on the chips to determine whether fabrication errors have occurred. For example, a single bit in a register (a storage location within the FPASP5) could be "stuck at zero" or "stuck at one". This error does not indicate a design flaw, rather an error in fabrication. Fabrication errors can occur for several reasons. One of the most common causes of a fabrication error occurs during the lithography process. When the layout (design) of a chip is transferred to a silicon wafer, the image of a microscopic dust particle may also be transferred. Since dust particles are significantly larger than the circuit pathways of a chip, the dust particle's image can easily short two pathways together.

In order to initiate this process, the test programs needed to be designed and simulated on the VHDL model of the FPASP5. First, A GNU emacs editor was used to write a test program. Then the test program was assembled using a

GNU assembler modified for FPASP5 assembly language. Finally, the program was executed on the VHDL model of the FPASP5, running on a SPARC-20 workstation. Output files from the VHDL simulation were then examined to confirm that the program ran as intended. Output files include register dump files, memory dump files, and car output files. The register dump files contain the value saved in each register at every clock cycle. The memory dump files contain the values stored at each address in RAM (memory element contained within the WSSP). The car output file shows which microcode instruction was being executed at a particular time. These output files, along with several others, are used to debug the assembly language program if necessary.

Several test programs were created in order to test the functionality of the FPASP5 completely. The first test that was written was designed to test each 32 bit register. This was one of the most important tests to develop because registers are used in every single assembly language program written. The register test program begins by defining two constants. AV1 is defined as 0xAAAAAAAA and AV2 is defined as 0x55555555. The "0x" tells the processor to expect a hexadecimal number. When converted to binary notation, these constants are 32 bit numbers. 0xAAAAAAAA is equivalent to 1010 1010 1010 1010 1010 1010 1010 1010 and 0x55555555 is equivalent to 0101 0101 0101 0101 0101 0101 0101 0101. By using these patterns of alternating zeros and ones, the test program can determine whether or not two bits have been shorted together. This test program will also test for "stuck at" errors.

Within each test, their are specific tests called subtests. In this case, within the register test, their is a subtest for each register. Basically, the same evaluation is performed on each register. A counter that points to the current subtest is incremented. AV1 is loaded into the register. The value in the register is then compared to AV1. The only reason for these two values to differ is a defective register.

Next, a branch statement tests for a specific condition and then executes another section of code if that condition is true. In this case, the branch statement tests for the equal condition. The equal condition or unequal condition has been previously set up by the compare statement. If the two values were equal, then the program executes to the next test. However, before that happens the next statement after the branch will execute. Immediately following the branch condition, their is a statement that stores the value of the register in memory. This is done so that the error can be examined when the actual chip is being tested.

If the two values were not equal, then the program will branch to a trap handler. A specific trap handler was written just for these tests. The trap handler counts the number of errors that have occurred and prints out a message telling researchers which test and which subtest have failed. The program will then execute the next subtest. After all subtests

that load AV1 into each register have been executed, all registers will be tested, in the same manner, using AV2 as the number that is read into each register.

A second register test was created in an attempt to shorten the amount of time that it took to completely evaluate every register. The second register test involves the use of a double precision floating point compare instruction. By using this instruction a pair of registers can be compared to another pair of registers.

In this test, the subtest counter was reset and then incremented in each subtest. In each subtest, four registers were loaded with AV1. A pair of registers can be used to hold a 64 bit number. One register will hold the 32 most significant bits, while the other holds the 32 least significant bits. One set of registers was then compared to the other set of registers. A branch statement was used to execute the next subtest if one pair of registers was found to be equal to the other pair. If the two were not equal, then the branch statement branched to the trap handler. The trap handler functioned exactly as it did during the first register test. Again, each register value was stored in memory so that the error could be examined during the actual test of WSSP. After all subtests that load AV1 into each register have been executed, subtests using AV2 will be executed.

The third test created was designed to test the arithmetic logic unit (ALU). First, test data, 0xAAAAAAAA, was loaded into two pairs of registers. Again, each pair held a 64 bit number. Next, a constant that defined the number of subtests to run was loaded into another register. This parameter was added due to the large number of subtests in this program. Researchers may not always want to run all 240 subtests. Also, it is important to understand that as the number of subtests executed increases, the utility, or usefulness, of each subsequent test decreases.

A short C program was used to encode the actual machine instructions that test the ALU. These instructions were then listed in a table. In order to call the first machine instruction, a jump command was used to go to the address in memory where the first instruction was located. In later subtests, the jump command is used to go to the address of the first instruction plus an offset. Since each instruction is located eight bytes apart in memory, jumping to the address of the first instruction plus an offset of eight bytes will, in effect, jump to the second instruction. Using an offset of 16 bytes will cause the program to jump to the third instruction. The next instruction immediately following the jump command will execute. In this "delay slot" there is another jump command that runs the next section of code. In essence, the program jumps to the table, performs an ALU operation using previously specified data, and then executes the next section of the test program. After this occurs, the result of the ALU operation is compared against an accepted value. The results are loaded into a register pair by the ALU operation, while the accepted value is loaded into another

register pair from a table of predetermined results. A branch statement is used to test for the equal condition. If the two register pairs are equal, then the program will repeat the entire process. Before this happens, a subtest counter is incremented, the value of the register pair that contains the results of the ALU operation is stored in memory, and the offset is incremented by eight so that the next subtest will run. If the two registers are not equal, then the program will jump to the trap handler and then continue with the next subtest.

The fourth test created was designed to test the incrementers. Incrementers are specific registers that are used to quickly increment the value stored in them by one. For example, an assembly language programmer could quickly perform the following operation: INC1 (first incrementer) = INC1 + 1. First, a constant that defines the number of tests to run is loaded into a register. Then the value to be incremented is loaded into the two incrementers accessible by an assembly language programmer from a table. The correct result (predetermined) is loaded into a register from another table. The value in the first incrementer is incremented by one. The value in INC1 is then compared to a the correct result. A branch command is used to test the equality of the value and the correct result. If the two are equal, then the same test is performed on the third incrementer. The second incrementer can only be accessed through FPASP5 microcode. If the two are not equal, then the program will branch to the trap handler and then test INC3. Once this subtest has been completed, the program will test the incrementers again using different data.

The fifth test created tests the functionality of the C pointer, the D pointer, the C incrementer, and the D incrementer. The C and D pointers are specific registers used for incrementing a value. However, an assembly language programmer can increment them by any value. The C and D incrementers are registers that store the value that the C and D pointers are to be incremented by. For example, if the C pointer had a value of 20 and the C incrementer had a value of 5, then an assemble language programmer could quickly execute the following operation: C pointer = C pointer + C incrementer. The value of the C pointer would now be 25.

In the program, a constant that defines the number of tests to run is loaded into a register. Next, the values that are to be incremented are loaded into the C pointer and the D pointer from a table. The correct result (predetermined) is loaded into a register. The C incrementer is then incremented (by one in this program, although the C pointer can be incremented by any value). The value of the C pointer is then compared with the register that contains the correct result. A branch statement is used to test for equality. If the two values are equal, then the same test is performed on the D pointer. If the values are not equal, then the program branches to the trap handler and then continues with the D pointer subtest. Once this subtest is completed, the program will test the C and D pointers and the C and D incrementers using

different data.

The next test was designed to test all branch conditions that are accessible through assembly language. A limited number of conditions had already been tested by this point. However, there were numerous other branch conditions that needed to be examined in order to completely test the branch command. In this program, every branch condition was tested twice. In the first test, registers are loaded, compares are preformed or other operations are executed so that the branch condition is true. A branch statement is then used to test for the condition. If the condition is true then the program would execute the second test. The only reason that the branch condition could be false is if there is a fabrication error or an error in the FPASP5 model. Otherwise, the program will branch to the trap handler and then continue to the second test.

In the second test, registers are loaded, compares are performed or other operations are executed so that the branch condition is false. A branch statement is used to test for the condition. If the condition is indeed false, then the program will not branch. A jump statement after the branch will cause the program to go to the next subtest. However, if the condition is true, indicating an error, then the program will branch to the trap handler and then continue with the next subtest. As always, the subtest counter is incremented before the next subtest is executed.

The final test is designed to test other assembly language commands. First, test data is loaded into several registers. Then a specific operation or command is executed and the answer is stored in a register. A predetermined result is loaded into another register from a table. The two registers are then compared. A branch statement is used to test for equality. If the two values are equal, then the program executes to the next test. If they are not equal, then the program branches to the trap handler before continuing to the next test. Of course, before the next test is executed, the subtest counter is incremented and the answer is stored in memory for debugging purposes.

In addition to the test programs, out research team needed a way to quickly execute an individual test or multiple tests. A flexible program in which each test is designated as a subroutine was written to meet this need. In the main routine, the user specifies which tests he/she plans to run by changing a 32 bit constant. The main routine will shift the constant by one place. A branch statement is used to check the shifted out bit. If the shifted out bit is equal to a one then the program will go to a section of the main routine that calls the proper test. In this section, the memory address of the first test is read from a table. A jump command is used to execute the test that begins at that memory address plus an offset. After the test is executed, the test counter is incremented and the offset is incremented. If the shifted out bit is a zero, then the offset and test counter are incremented. The process

repeats until the 32 bit number has been shifted the maximum amount of times. This maximum value is equal to the number of subroutines in the program.

For example, if a user wanted to run test zero and test two then he/she would specify the following 32 bit number 0000 0000 0000 0000 0000 0000 0000 0101. When the number is shifted by one place, it will become 0000 0000 0000 0000 0000 0000 0000 0010 and the shifted out bit will equal one. The branch condition will test for this condition, and jump to the section of the main routine that calls the proper test. In this section of code, the address of the first test is read from a table, and a jump command is used to execute the test. After the first test is executed, the offset is incremented by four, the test counter is incremented by one and the 32 bit number is shifted again. This time the shifted out bit equals zero. The offset is incremented by four and the test counter is incremented by one. The 32 bit number is then shifted a third time and the shifted out bit equals one. The branch statement cause the program to go to the section of code where the proper test is executed. Here the address of the first test is read from a table. A jump command will execute the test that begins at that address plus eight. In other words, the second test will execute. This process will repeat an until the maximum number of shifts, seven have occurred.

Results

A small number of minor bugs were discovered in the VHDL model of the FPASP5, the FPASP5 assembler, and in FPASP5 documentation as a result of writing as a result of engineering test program for the FPASP5. First, the register pair R13 is not recognized by the assembler. The error was caused by an incorrect register definition and will be fixed when the assembler is updated. The next two errors involve branch conditions. UFPPZ, one of the conditions that checks for equality, according to the documentation, is not valid according to the assembler. As a result of this bug, the condition words were changed to be more user friendly. The equivalent condition, UFPEQ, is now recognized by the assembler. Next, according to the FPASP5 assembly language instruction set documentation, the condition USO should test for Upper Shifter's shifted out bit equal to zero. However, the documentation should say that USO will test for Upper Shifter's shifted out bit equal to one. Finally, the FPASP5 instruction that is supposed to subtract one double precision floating point number from another actually adds the two floating point numbers together. The same bug was found in the single precision version of the same command. This bug was caused by an error in the VHDL model and was corrected shortly after it was discovered. The test programs also discovered several bugs in the logic-level model of the FPASP5. However, this was expected since the model is fairly new and is still undergoing evaluation. As of this

writing, the actual WSSP chips have not been manufactured. However when they are, this battery of tests will be extremely useful in determining the functionality of the WSSP chips.

Conclusion

In conclusion, the WSSP is an adaptable and highly efficient processor that is on the cutting edge of today's technology. The processor is designed to quickly handle floating point calculations and complicated vector operations. In addition to the outstanding performance of the WSSP, it also has a small size, low weight, and low power consumption, making it ideal for advanced signal processing applications.

In order to design the WSSP, several software models of the FPASP5 were created. The first model of the FPASP5 was created using VHDL and was used to debug the FPASP5 assembly language. The second model was created using IRSIM an event-driven logic-level simulator and was used to debug and optimize the FPASP5 hardware design. In order to effectively debug the FPASP5 software and hardware several test programs were developed. These programs were designed to completely test the functionality of the VHDL model, the IRSIM model and the actual WSSP chips. Currently there are test programs designed to examine all registers, the arithmetic logic unit, the incrementers, the C and D pointers, the C and D incrementers, all branch conditions that are available to an assembly language programmer and other assembly language commands. In addition, a flexible program was written to allow an individual test or multiple tests to be executed. As a result of this project, a small number of minor bugs were discovered in the VHDL model of the FPASP5, the FPASP5 assembler, and in FPASP5 documentation. Several bugs were also discovered in the logic-level simulation. Once the hardware design is debugged and optimized, the WSSP chips will be manufactured and tested for fabrication errors.

F-16 EM SCATTERING

Sharleen P. Johnson

New Hartford High School

33 Oxford Rd.

New Hartford, NY 13413

Final Report for:

High School Apprentice Program

Rome Laboratory

Sponsored by:

Air Force Office of Scientific Research

Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

F-16 EM SCATTERING

Sharleen P. Johnson
New Hartford High School

Abstract

The EM (electromagnetic) scattering off an 1/32 scale F-16 was studied, using an IR (infrared) measurement technique. Carbon loaded Kapton paper, sensitive to EM radiation, increases in temperature in a direct proportion to the intensity of the radiation. An IR camera, tied in to AGEMA Thermovision software, detects the temperature on the paper and allows this information to be processed. The model F-16 was placed into a hole in the paper, radiated with EM radiation from a horn antenna, and the EM scattering was measured using an IR camera. This is a thorough, time and cost efficient method of measuring EM scattering.

F-16 EM SCATTERING

Sharleen P. Johnson

Introduction

Recently, a new and more efficient method has been developed for the making of EM (electromagnetic) measurements. With an IR (infrared) camera setup, an entire slice of continuous EM data can be obtained in a rapid and minimally perturbing manner. Previously, probes were used to determine EM field strength, but this was not an ideal way to validate a computer model because the EM levels in the spaces between the probes had to be arrived at either by a thorough and time-consuming use of probes or through interpolation. Also, the probes themselves caused some scattering, so even the known points of EM density may not have been completely accurate. The IR measurement technique has made it possible to easily collect data over a two dimensional plane when measuring EM fields and scattering effects. When this methodology became accepted as an accurate measuring tool, the IR camera setup began to be used to measure the EM scattering off of simple objects. This method has now been used to test the EM scattering off of more complex objects, in this case a 1/32 scale F-16. These variables were used: orientation of the plane in relation to the horn antenna, position of the plane in the paper, and frequency.

Methodology

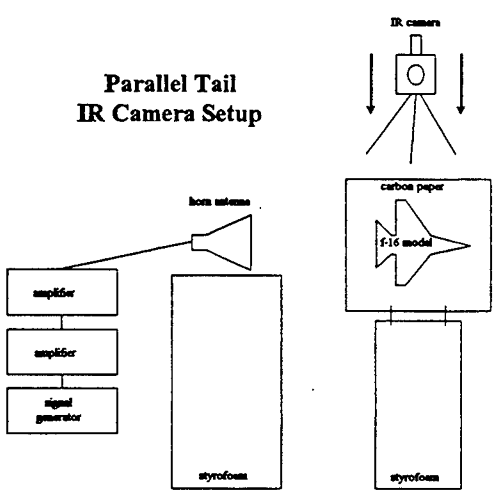
To analyze (or detect) the EM field densities, an IR camera setup was used. This setup consisted of a horn antenna, a sheet of carbon loaded Kapton paper of known resistance (in this case, the resistance = 1500 ohms), and a long wave (8-12μm) IR camera. When EM radiation is emitted from the antenna and passes through the carbon paper, the paper's temperature rises above the ambient temperature in an amount directly proportional to the strength of the EM field. The paper then re-radiates this heat energy as radiation of a different wavelength, which is picked up by the IR camera and displayed on the computer screen as a two dimensional temperature map.

Before radiating each time, the baseline temperature situation is recorded. Then the RF generator and the amplifiers for the horn antenna are turned on, and the temperature of the carbon loaded Kapton paper rises as it is exposed to the EM radiation. A pattern begins to form, representing the scattering of the EM energy, and the temperature is allowed to stabilize before that picture is recorded. The baseline (or ambient) temperature is then

subtracted from the temperature resulting from the RF heating, and the resulting data is normalized and plotted.

The F-16 model, which is plastic, was sprayed with a special silver paint to allow conductivity over its surface. Several coats were applied to obtain optimal results (high conductivity), and a final layer of non-conductive paint was applied to protect the silver paint. The canopy of the model was covered with aluminum tape and attached separately so that it could be easily removed in later trials.

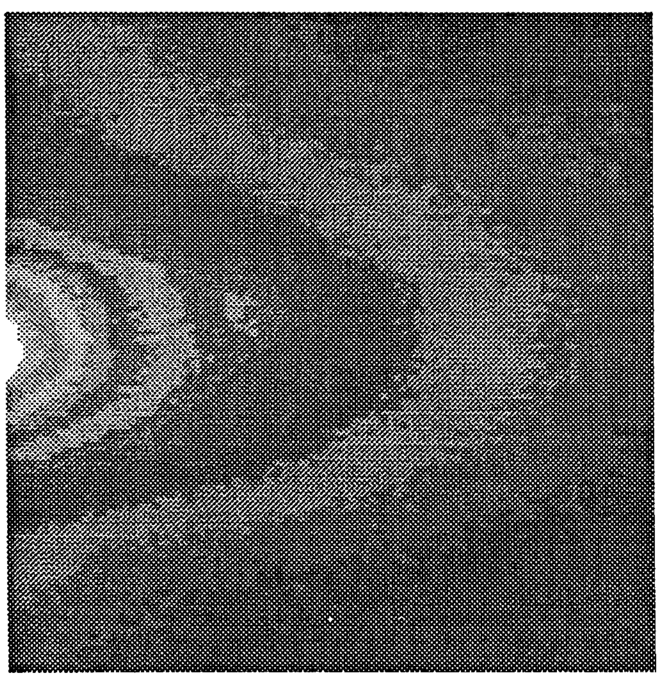
There were four different positions in which the airplane was examined: the parallel nose, wing, and back views, and the perpendicular front view. The terms 'parallel' and 'perpendicular' refer to the orientation of the plane in the paper. The parallel position of the F-16 was arranged so that the wings lied on the same plane as the paper. A hole was cut into the paper so that the model can be placed into it and the EM data collected will correspond to the fields reflecting off of a cross section of the model extending from the nose to the tail. A piece of styrofoam board, which minimally affects the EM scattering, has been attached to the paper to provide support and stability. The antenna is positioned at the side of the paper, and the IR camera has been placed in front of the paper.



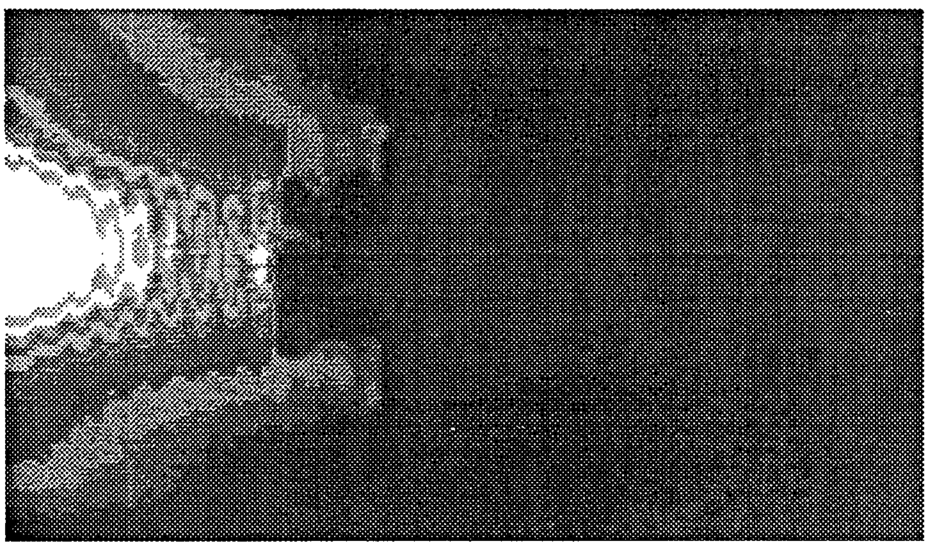
The perpendicular position of the F-16 has been arranged so that the nose and tail protrude from opposite sides of the paper, allowing the EM scattering from a nose on position to be examined. The horn was positioned directly in front of the paper, and the IR camera was also in front of the paper, but slightly off axis so that the antenna would not obscure the picture. Each position was examined at 5 GHz. Also, the parallel nose view was examined at 3 GHz. All measurements were taken in an anechoic chamber so that only the experimental EM fields affected the data.

Results

1. Baseline one- The horn antenna is radiating at the side of an intact carbon sheet, and the IR camera is positioned to view the broad side of the paper. The temperature, and therefore the intensity of the EM field, is the most powerful at the left side of the paper and dissipates from that point. As a result of convection, a warmer temperature pattern has spread to the upper portion of the paper as compared to the lower portion. This is a phenomenon that will be seen throughout all of the pictures in this experiment.

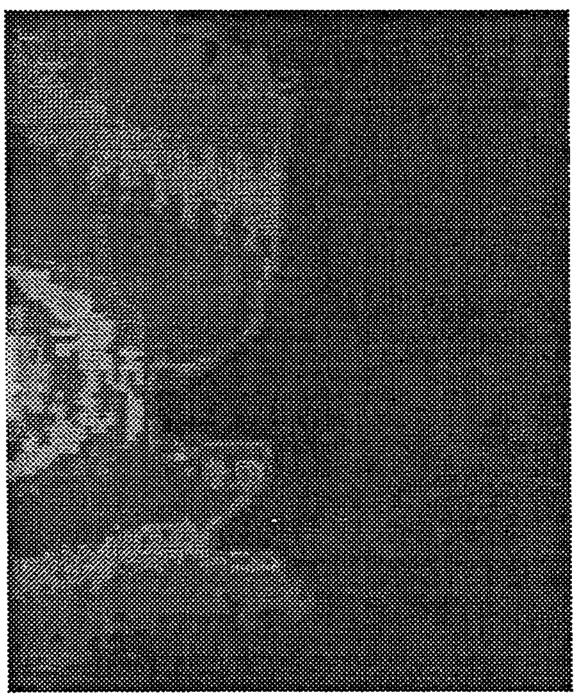


A. <u>Parallel tail</u>- The carbon paper has been cut so as to place the model F-16 within it. Therefore the EM field data collected will concern a two dimensional slice along the center of the model, not the edge. The radiation is reflecting off of the tail and wings back towards the source, resulting in a standing wave. A shadow has appeared in front of the plane, although a small amount of radiation appears to have been conducted over the surface of the plane and shows up as a slightly higher than ambient temperature along the edge of the nose.

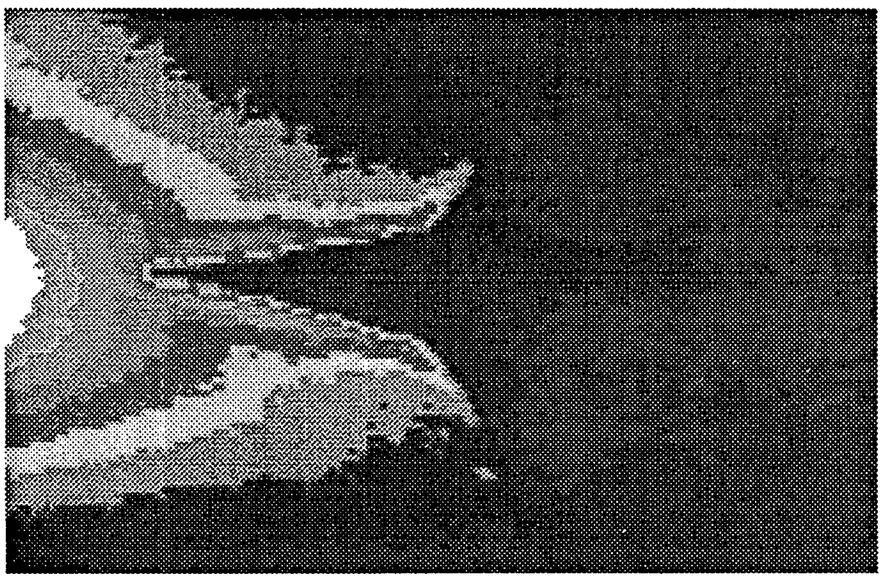


B. Parallel wing- The paper has been turned upon its edge, with the side of the plane now being the focus of the radiation. A standing wave has once again appeared, seen as a ripple effect to the left of the plane. A few small warm spots have developed behind the wing and at the base of the tail. The warm spot at the wing corresponds with the location of three small protrusions on the model F-16, which suggests that the scattering off of these objects is the cause of the temperature differential. The disturbance at the base of the tail is most likely the result of a small tear in the carbon loaded Kapton paper at this position.

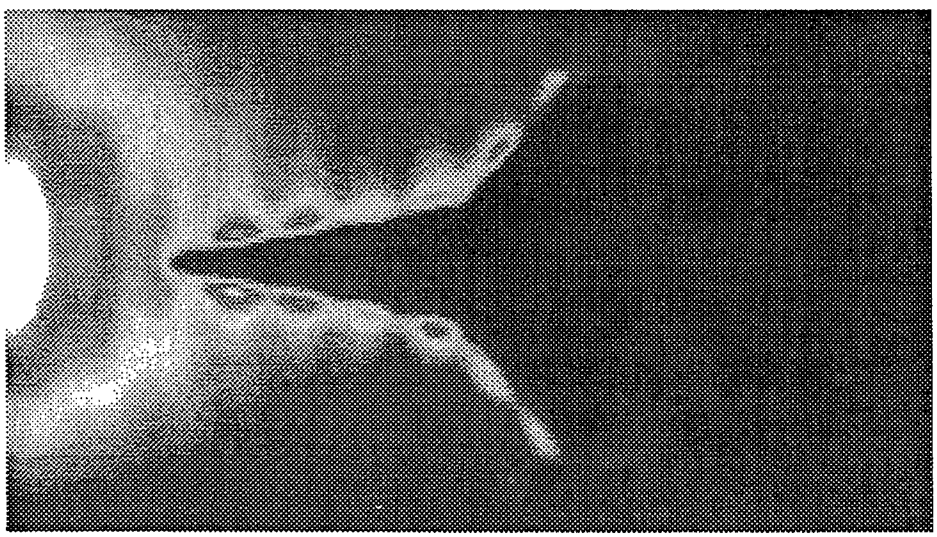
The lower right hand portion of the model appears to be lying in an electromagnetic shadow, but analysis of the baseline temperature shows that this area had been cooler than the other areas previous to the emission of the EM radiation, and has risen in temperature at a comparable amount in relation to the surrounding regions. The left hand edge of the model F-16 has heated up above the ambient temperature. A logical assumption would allow the conclusion that it is not only because of surface resistance to the EM radiation, but also a normal transfer of heat from the warm paper to the relatively cool model.



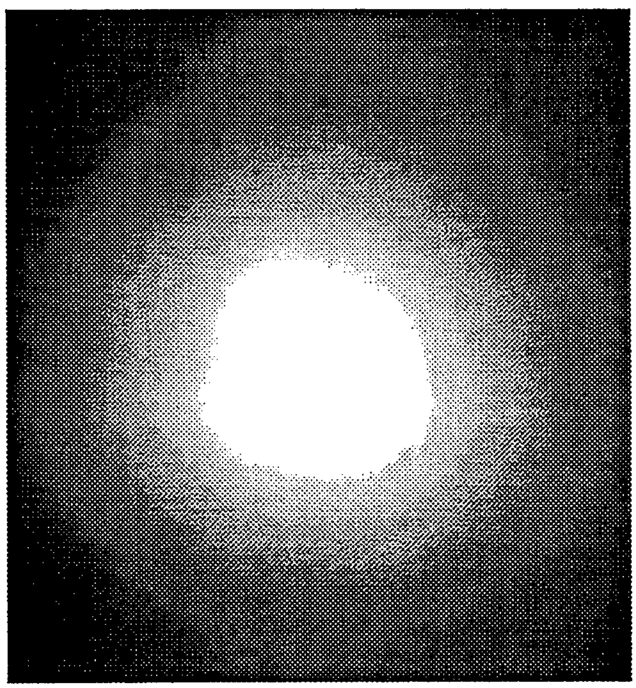
C. Parallel nose- The paper has been rotated another 90 degrees so that the EM radiation is now focused upon the nose of the model F-16. The warmer temperature zones are located along the nose and continue until the wing base, dissipating as the distance from the nose increases. As before, the heated paper has warmed the adjacent edges of the plane to a temperature slightly above the original baseline values. A cooler region is located behind the plane, which apparently was protected from EM radiation by the model.



D. Parallel nose 3GHz- The paper has remained in the same position, but the frequency of the EM radiation has been changed from 5GHz to 3GHz. The new wavelength has caused a definite change in the temperature pattern along the nose. Six hot spots have appeared on each side, becoming smaller and more spaced out as they progress towards the base of the wings. A standing wave has also emerged, which was not observed in the original 5GHz situation. Warm temperatures outline the missile launchers, and the area behind the tail of the model is shaded once again, protected from the EM fields by the plane.

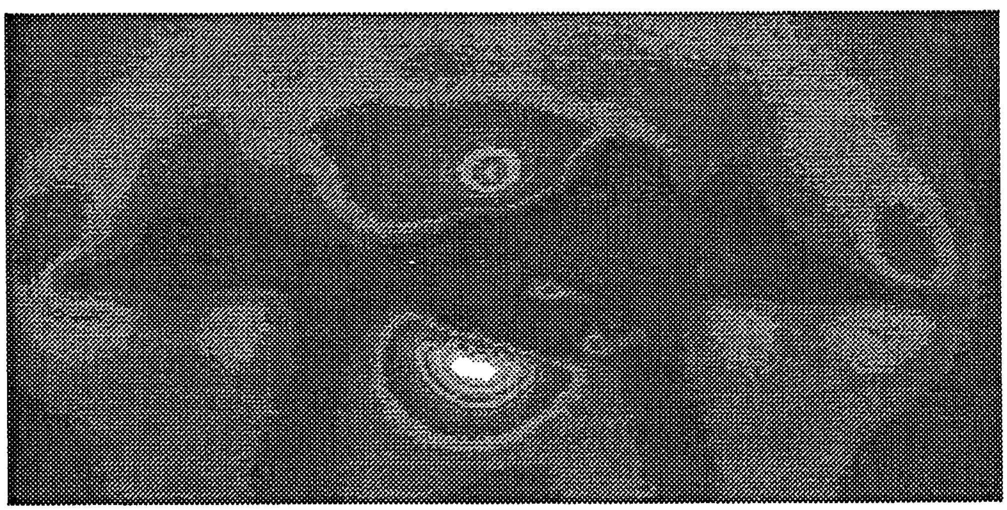


2. <u>Baseline two</u>- The horn antenna is radiating at the front of an intact carbon sheet, and the IR camera is positioned to view the broad side of the paper. The temperature, and therefore the intensity of the EM field, is the most powerful at the center of the paper and dissipates from that point. As a result of convection, a warmer temperature pattern has spread to the upper portion of the paper as compared to the lower portion. This is consistent with the other baseline pattern.



A. Perpendicular front- The body of the plane is perpendicular to the flat section of the carbon loaded Kapton paper, and the nose and most of the wing portions are on the front side. The horn antenna is positioned directly in front of the paper, but the IR camera is slightly off axis so as to keep the picture unobscured by the antenna.

Hot spots are located above and below the central portions of the model's body. Warm spots have formed at the tips of the wings, and slightly cooler areas are found below the central parts of the wings. Cool areas extend down from the meeting place of the wings and fuselage, and up from the central part of the wings. This is a relatively complex pattern, one that would probably be quite difficult to determine through the use of probes, yet is quite simple to obtain through the IR camera setup.



Conclusion

The measuring of EM scattering through the use of IR measurements is a reliable, time and cost efficient technique. Detailed patterns, such as standing waves and hot spots, are quickly and easily detectable. The researcher, unlike when using probes, can take a whole two dimensional slice of data at once, rather than measuring only a single location at a given time.

The temperature data can be dealt with in spreadsheet form. The ambient temperature can be subtracted from the experimental results and then the resulting temperature can be normalized, allowing direct comparison with simulated data. Rome Lab is currently working on a GEMAX simulation for EM scattering off a 1/32 scale F-16, and the data obtained in this experiment will then be used to validate the results.

The IR camera setup is a viable substitute for other far more costly methods of measuring the EM scattering off of objects.

References

1. Pesta, A.J., and Seifert, M.F. "Infrared Measurements of Electromagnetic Fields", DUAL-USE Technologies and Applications Conference, SUNY Institute of Technology, Marcy, NY, May 1993.

2. Norgard, J.D., Metzger, D.W., and Sega, R.M., "Analysis of Probe Measurement Accuracies and Scattering Effects", Technical Report RL-TR-91-252, Rome Laboratory/ERST, Griffiss Air Force Base, NY.

HYPERTEXT MARKUP LANGUAGE

Michael J. Manno

Frankfort Schuyler High School
Palmer Street
Frankfort, NY 13340

Final Report For:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

HYPERTEXT MARKUP
LANGUAGE

MICHAEL J. MANNO
FRANKFORT SCHUYLER
HIGH SCHOOL

ABSTRACT

This paper describes the work done during the simmer of 1995 on developing World Wide Web pages to document the contribution of the Design and Diagnostics Branch of the Rome Laboratory mission.

HyperText Markup language or (HTML), is the language used to develop and set up World Wide Web pages on the internet. To create HTML documents, numerous tags are used to specify the command. The tags discussed in this paper are the most common tags that will be found in almost all HTML documents.

INTRODUCTION

The Rome Laboratory Simulation and Modeling Branch mission is to develop the tools and methodologies to improve the reliability and availability of electronic systems. The Simulation and Modeling Branch mission is supported by Electronics Design Group and the Integrated Test & Test Automation Group. Together the Design and Test Groups coordinate and develop techniques for simulation and modeling to support an integrated design process. With the development and design of increasing technology enables the Air Force to reach the goal of Global Reach and Global Power by allowing two-level maintenance of operational systems and enabling the development of a highly mobile and supportable force structure.

METHODOLOGY

In recent years, the internet has grown at a very rapid rate. The use of the World Wide Web has brought instantaneous information for people throughout the world. The growth of homes with computers from December of 1994 until May of 1995 has increased 4 percent. Also, homes with on-line services has increased to 3 percent, which is predicted to increase another 4 percent by December of 1995. Soon every house will be connected to on-line services and products such as the newspaper will be obsolete.

In order to create HyperText Markup Language, or HTML documents, various tags are used. Tags usually open with < > angle brackets along with </ > angle brackets which ends a command.

example: #1 <TITLE> this is the title </TITLE>

HTML is case insensitive; therefore the first example could be expressed as:

example: #2 <title> this is the title </title>

Although HTML is case insensitive, tags are usually written in all upper case to make them stand out more.

Below is an example of a HTML web page documenting the Analog Hardware Description Language being developed at Rome Laboratory.

```
<HTML>
<HEAD>
<TITLE>ANALOG HARDWARE DESCRIPTION LANGUAGE</TITLE>
</HEAD>
<BODY>
<CENTER>
<A HREF="/source/ERDD-fact-sheets.html">
<IMG SRC="/images/erddhead.gif"></A>
<HR SIZE=2>
<H2>ANALOG HARDWARE DESCRIPTION LANGUAGE</H2>
<IMG SRC="/images/ahdl.gif">
</CENTER>
<IMG SRC="/images/er-bar.gif">
<H3><I>DESCRIPTION:</H3></I>
The Analog Hardware Description Language program is addressing the lack
of <I>standard</I> industrial methodologies for capturing and documenting
design information necessary for the life cycle support of analog and
mixed mode systems. The development of this language is being coordinated
within the Institute of Electrical and Electronics Engineers (IEEE)
under the auspices of the Computer Society. The <I>objective</I> of this
effort is to develop a standard Analog Hardware Description Language to
describe the physical design, electrical behavior, logical structure and
systems annotation information for analog and mixed-signal systems. This
standard is being developed be extending the VHSIC Hardware Description
Language (VHDL) to support the description and simulation of continuous
time behavior (commonly called analog) and mixed continuous/discrete time
behavior (commonly called analog/digital).
<HR SIZE=2>
<H3><I>PAYOFF:</H3></I>
<UL>
<IMG SRC="/images/rl-ball-yellow.gif"> Reduced Life Cycle Support Costs<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Hierarchical Analog Behavioral Modeling
and Simulation Environment<BR>
```

```
<IMG SRC="/images/rl-ball-yellow.gif"> Rapid, Top-Down Analog Design<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Portable and Technology Independent
Designs<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Formalized Design Documentation<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Design Sharing and Reuse<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Affordable Low-Cost Designs<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Amortization of Tool Costs Across the
Industry<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> IEEE Industry Standard Developed<BR>
</UL>
<H3><I>APPLICATIONS:</H3></I>
<UL>

<IMG SRC="/images/rl-ball-yellow.gif"> Complete Analog and Digital System
Hardware Modeling and Simulation<BR>
<UL>
<IMG SRC="/images/rl-ball-yellow.gif"> High Level Analog Modeling<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Direct Specification of the Behavior
and Structure<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Design Verification<BR>
</UL>
<IMG SRC="/images/rl-ball-yellow.gif"> Life-Cycle Support<BR>
<UL>
<IMG SRC="/images/rl-ball-yellow.gif"> Maintenance<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Upgrades<BR>
<IMG SRC="/images/rl-ball-yellow.gif"> Reprocurements<BR>
</UL>
</UL>
<IMG SRC="/images/er-bar.gif"><P>
<CENTER>
<PRE>
<ADDRESS>
POC: <A HREF="/names/drager.html">Steven L. Drager</A>     Updated 28 July 1995
/  Expires 28 July 1996
</ADDRESS>
</PRE>
</CENTER>
</BODY>
</HTML>
```

Above is an example of a HTML document typed at Rome Laboratory during the summer of 1995

The html document consists of two sections, the header and the body. The <HEAD>,and </HEAD> tags surround the header information which typically consists of a document title and other global configuration. The <BODY>, and </BODY> tags surround the body of the document. This is where the text and graphics of the document appear. Body tags typically consist of formatting tags and references to images and other documents. The <IMG SRC="..."> tag is used to display graphic images. The <A HREF="...",</A> tag is used to generate a hyperlink to another file.

The center tag <CENTER> surrounds five other tags. Everything after the center tag and Before the </CENTER> tag will be centered.

The <A HREF="/source/ERDD-fact-sheets.html">is hyperlink to another document or file. It is possible to link to another document or file in other directories by stating the path from the current document to the linked document. Example, a link to the file ERDD-fact-sheets located in the sub-directory source would be: <A HREF="/source/ERDD-fact-sheets.html">

Since pictures can not be generated in HTML, they can be inserted in various places. Therefore, the <IMG SRC="/images/erddhead.gif"> will now insert the picture erddhead from the images directory.

<HR SIZE=2> is a horizontal rule. It is a line that will appear to make the document more structured and legible. The size will be increased to 2 percent..

The size of the text could also be increased. The <H2> tag before ANALOG HARDWARE DESCRIPTION LANGUAGE will appear larger than the rest of the text. The style of the text could also be changed by adding the <I> tag which will make the text italicized.

When typing a list where order is not important, the unordered list tags may be used. Placing the <UL> tag, an unordered list is created. Therefore. each additional list item <LI>, the text is listed, unnumbered, with bullets before each. However, in this document, list items were not used. Instead, the yellow ball picture was used for bullets. The
 tag or line break causes no spaces to be inserted between list items.

In displaying HTML documents multiple spaces between words is ignored, as are line breaks. Therefore, in order to create more than one space, the pre-formatted text tag or <PRE> is used. In this document it was placed before the point of contact to make the document more appealing.

It is also possible to make text stand out more by using bold text <BOLD>. Also to underline text the <U> tag is used.

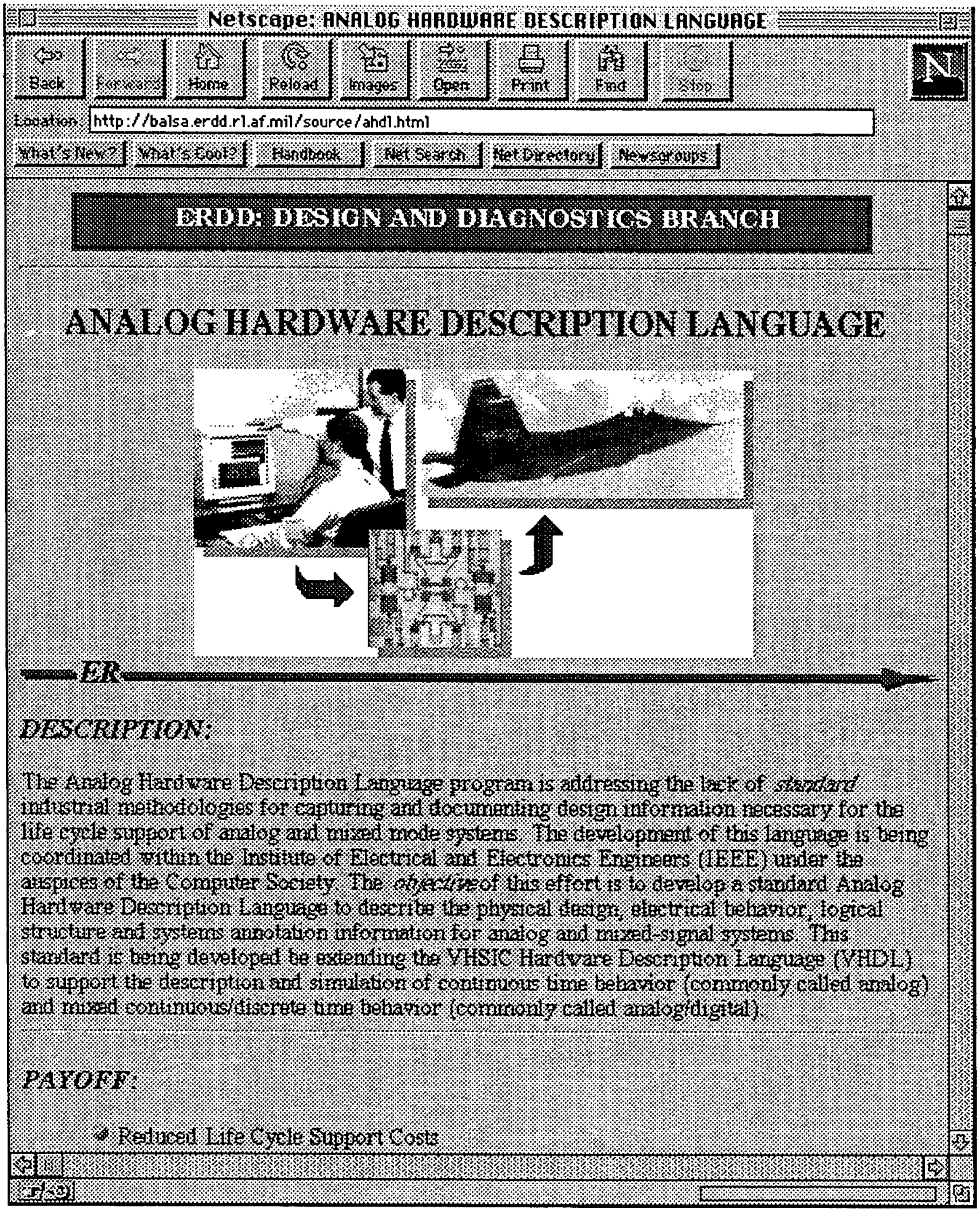


Figure 1.1 Netscape browser of AHDL page.

CONCLUSION

Throughout the apprenticeship program, many tools and skills were learned. Microsoft word and various programs were used for the first time. Numerous tasks were accomplished including the development of a home page for ERDD. The page included fact sheets describing projects, also with personnel data sheets describing the people in the branch, along with contact information. In addition, pictures were scanned in of various people and their projects. Overall, during the apprenticeship, the HTML language and various tools that were learned could now be used at ease.

REFERENCES

1. "Producing HTML Documents."
   http://www.bham.ac.uk/documents/production.html

2. "The HTML Language."
   http://union.ncsa.uiuc.edu/hypernews/get/www/html/lang.html

3. "NCSA HTTPD Overview."
   http://hoohoo.ncsa.uiuc.edu/docs/overview.html

4. "HTML Background Selector."
   http://www.imagitek.com/bcs.html?

STUDY IN COMPUTER APPLICATIONS AND NEURAL NETWORK PROCESSES

Kathleen D. Scheiderich

Rome Catholic High School
800 Cypress Street
Rome, NY 13440

Final report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.

And

Rome Laboratory

September 1995

TABLE OF CONTENTS

STUDY IN COMPUTER APPLICATIONS AND NEURAL NETWORK PROCESSES

Kathleen D. Scheiderich
Rome Catholic High School

Abstract

I accomplished two projects during the 8-week tour at Rome Lab's Electronic Intelligence Developement Facility (EDF). The first enterprise was to design a conference room and detail structural changes and renovation. The second project was to study and apply neural network techniques to an image pattern-matching problem. Both projects required comprehensive use of computers and various software.

STUDY IN COMPUTER APPLICATIONS
AND NEURAL NETWORK PROCESSES

Kathleen D. Scheiderich

1.0 Introduction

This paper will discuss two separate areas, Architectural Design and Neural Networks.

The section covering Architectural Design will discuss the methods used for creating floor plans, developing the new room and instituting the desired changes in the design.

In the section regarding Neural Networks, the science, functionality, and application of Neural Networks will be discussed.

2.0 Architectural Design

The focus of this project was to create a 'blueprint' of a conceptual conference room within the parameters of an existing space. It was necessary to first produce the layout of the existing rooms, and then to visualize and experiment with the best arrangement of the new area, accounting for certain variables.

2.1 Creating the Original

In order to create an accurate picture of the area, measurements of all walls, doors, windows, and floor space were taken. Using Claris* MacDraw II 1.1 on a Macintosh IIfx, all of the walls, and other room features were then drawn in. In order to facilitate viewing, all diagrams were scaled down to .25 inches per foot. A grid, 2 feet by 2 feet (.5 inches x .5 inches) was also used in all diagrams.

2.2 Developing the New Plan

When the existing room was drawn and scaled, it was then necessary to implement the desired changes. What was wanted was a conference room, with dimensions of approximately 18 feet by 21 feet, with seating for 18 people, and room for a table 8 feet long. The room could also not use all of the available space. A six-foot-wide hallway had to be left, running lengthwise. So the conference area, therefore, had to also be placed lengthwise. This was deemed the best utilization of the space. Figure 2.1 below is the room before any modifications, and Figure 2.2 shows the same area after the renovation.

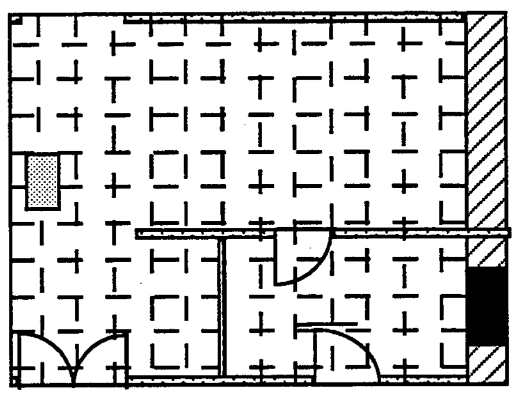


Figure 2.1- Before the changes.

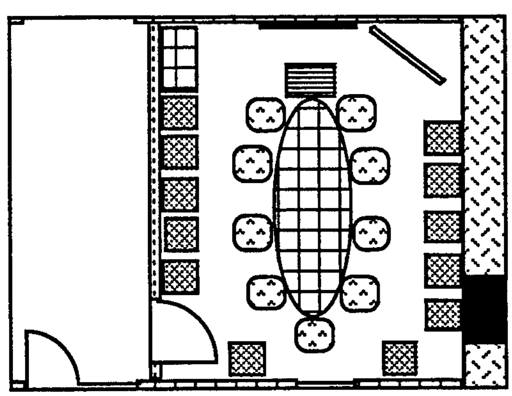


Figure 2.2- After the changes.

3.0 Neural Network

The focus of this project was to become familiar with the usage and technology of Neural Networks. This was accomplished with the help of three sources, the book *"Comparison of Neural Network and Statistical Techniques for Radar Pulse Detection"* {2}., *NeuralWorks Explorer* {3}, and the InterNet and World-Wide Web services. These sources were of invaluable use in learning many different facets of Neural Network technology.

3.1 Study of Neural Networks

The study of Neural Networks was begun by reading *"Comparison of Neural... "* , which provided all of the information regarding their history, early attempts, biological models (i.e. the brain, neurons and nervous system), and current uses and developments. The software program *NeuralWorks Explorer* was run on a PC and provided an opportunity to actually see the technology work. The InterNet and World Wide Web services provided supplementary information on historical and current uses.

3.2 Applying Neural Network Technology

Using the *NeuralWorks Explorer* the technology was applied to a recognition problem involving 4 images. However, before this application is discussed, it might be beneficial to describe Neural Networks in more detail.

3.2.1 Description of Neural Networks

A Neural Network is a type of artificial intelligence, trained to

recognize and learn data. There are many different paradigms, or types, of networks. In all paradigms, there is an input layer, where information is applied. however, differences then appear. The network which was used in this experiment was a Back-Propagation network. This type has one or more hidden layers and an output layer, and is one of the more common paradigms. Neural Networks, in general, are a technology meant to emulate the brain and neurons, most especially in the way that it learns data. The neurons of a physical brain are fired, or excited, by electrical impulses, or stimuli, creating a response. When that stimuli occurs again, the neurons excited at last occurrence will fire again with a stronger response, because they have "learned" that impulse. Neural Network "neurons" have weighted connections which are adjusted upon firing to produce a stronger response on another recognition of same impulse {2}.

The Back-Propagation network used in this project runs by propagating information forward to the output. However, information is also fed back to the input.

3.2.2 Image Recognition

The Back-Propagation network was given a set of statistics, pertaining to each of four different images. These statistics used data such as length of specific parts of each image. This data was fed into the input, and propagated through the training set. Two trials were made, and results compared. The first trial used a training set of 200 times, and the second trial a set of 100 times.

The error was sufficiently greater in the smaller set to be able to theorize that as the training length increases, error decreases, to the point were it would no longer be significant.

4.0 Conclusion

As a conclusion, it is necessary to state that I learned many things, particularly pertaining to computers. Before I worked here, I was apprehensive about experimenting with them. However, in my time here, I learned how to use many programs and real-life applications, as well as basic skills. I plan to build on the knowledge I have gained, and hope that it will be auspicious in future.

REFERENCES

1.) *MacDraw II* . 1987-1989. Claris Corporation

2.) Richard V. Cornish Jr. *Comparison of Neural Network and Statistical Techniques for Radar Pulse Detection* . August 1992.

3.) *NeuralWorks Explorer* . NeuralWare, Inc; Pittsburg, PA. 1991

THE CONFIGURATION AND USE OF THE LAB-PC+ BOARD

Heidi M. Schwartz

Camden Central High School
Mexico St.
Camden, NY 13316

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

THE CONFIGURATION AND USE OF THE LAB-PC+ BOARD

Heidi M. Schwartz
Camden Central High School

Abstract

A data acquisition board was installed and configured in a personal computer. The previous was done to provide a data acquisition tool for electro-optic research at the Rome Laboratory Photonics Center. A program was written in a symbolic language environment (LabVIEW) to control the data acquisition board in a user-friendly manner. An experiment was conducted to demonstrate simultaneous reading and writing of analog signals to electro-optic devices.

THE CONFIGURATION AND USE OF THE LAB-PC+ BOARD

Heidi M. Schwartz

Introduction

As each day passes, an increasing number of people are using computers and more and more uses of computers are being discovered. For example, in the beginning of the computer age computers were only used for typing material. Now look at what computers can be used for, everything from playing games, generating pictures, communicating to others through E-Mail, to even controlling complex laboratory experiments. From the aforementioned list, the latter was primarily focused on during this research with the National Instruments Lab-PC+ data acquisition board. The Lab-PC+ board which was installed allowed a computer to control laboratory devices.

Methodology

Before the program could be created that would allow both analog output and analog input to take place, the Lab-PC+ Board had to be installed and configured. Upon completion of the installation and configuration of the Lab-PC+ Board, the process of creating a virtual instrument that would control the board commenced. Virtual instruments were created by using LabVIEW for Windows, which is a program development application. LabVIEW is like many other programs, but is unique because of its graphics. Many programs are textual based rather than graphical, like LabVIEW. LabVIEW contains libraries of functions and development tools that are "designed specifically for data acquisition and instrument control." (LabVIEW Tutorial)

As time passed, it was found that creating a virtual instrument that would accomplish both analog output and input was not easy. The first step to creating this virtual instrument(VI) that would enable both analog input and analog output to take place was to have a VI that would cause analog input to occur. The Lab-PC+ Board had 50 different pins, each one of which had their own functionality. When dealing with analog input, pins 1-8 and pin #9, the analog input ground, applied. These analog input pins were going to be used for detecting the analog input signal from a light emitting diode. The correct analog input connections from the detector to the Lab-PC+ board were made (see fig. 1). Then the VI, *The one that is going to work.vi* (see fig. 2) that was used just for analog input was tested. After the correct connections and minor adjustments were made, the virtual instrument functioned correctly.

Then, analog output was looked at to the same degree as the analog input had been. It was found that the Lab-PC+ board only had two pins, 10+12 for analog output, and pin #11, analog output ground, that applied to analog output. The analog output was going to generate a signal that would in turn cause a LED to

light at a certain frequency and amplitude. The correct analog ouput connections were made from the Lab-PC+ board to the LED(see fig. 1). A test run was done and a few problems occurred. However, after making some subtle adjustments to the virtual instrument, *Function Generator.vi*(see fig. 3), that dealt with analog output, it was functioning in coherence with the board and LED.

In addition to the two previously mentioned VIs that were utilized with the PC+ board, a third one referred to as the *SubVI Compute Waveform.vi*(see fig. 4) was also employed. This particular program generates the type of signal desired. The *SubVI Compute Waveform.vi* works with the *Function Generator.vi* in determining the desired signal.

After being able to control the analog output and input functions through the use of the Lab-PC board, it was time to start making one VI, *The Final Episode of DAQ.vi*(see fig.5), that would do both analog output and input. Since two VIs had already been constructed that would allow analog output and input to take place, one VI was constructed which incorporated the other two VIs. Upon finishing the programming of this one VI by using the other VIs as SubVIs and the program LabVIEW for Windows, a test run was done. Similarly to the other VIs that had been constructed, *The Final Episode of DAQ.vi* required some adjustments too. After making the correct changes and adjustments the VI was functioning correctly.

Results

The Final Episode of DAQ.vi is a very intricate virtual instrument. This virtual instrument that was created has many abilities to make changes to both the analog output signal and/or the input signal(see fig. 5). For the analog output signal, one of the items that can be changed is the signal type. The signal type is capable of being one of the following four types, a sine wave, a square wave, a triangle wave, or a sawtooth signal. Also, the amplitude of the output signal can be changed. In regards to the analog input, the number of scans to acquire can be adjusted according to the desired amount.

Using *The Final Episode of DAQ.vi* I adjusted the amplitude of a sine wave signal to three volts and ran it(see fig. 6). Figure 6 shows the test set up parameters. Also, the voltage detected can be seen digitized on the potential difference graph. A small noise signal is apparent in the output graph.

Conclusions

The National Instruments Lab-PC+ data acquisition board is a low-cost multifunction analog and digital input board for the personal computer. The Lab-PC+ board has eight analog inputs and two analog voltage outputs. The Lab-PC+ allows for the communication of other laboratory devices through computer control. *The Final Episode of DAQ.vi* programs that was developed using LabVIEW for Windows allows for the adjustment of different analog signals to be output

and the virtual instrument allows for an adjustment to the input of the number of scans to acquire. The Lab-PC+ used in conjunction with the personal computer is an adaptable, costeffective board for laboratory test, measurement, and control. A program was developed which includes a user friendly graphical user interface for the sourcing of user-defined analog signal and for the simultaneous input of analog signals. The functionality of the combined data acquisition board and the LabVIEW program that was developed demonstrated by the simultaneous modulation of a light emitting diode and the acquisition of reading from a photodetector.

APPENDIX

ANALOG OUTPUT/INPUT WITH LAB-PC+

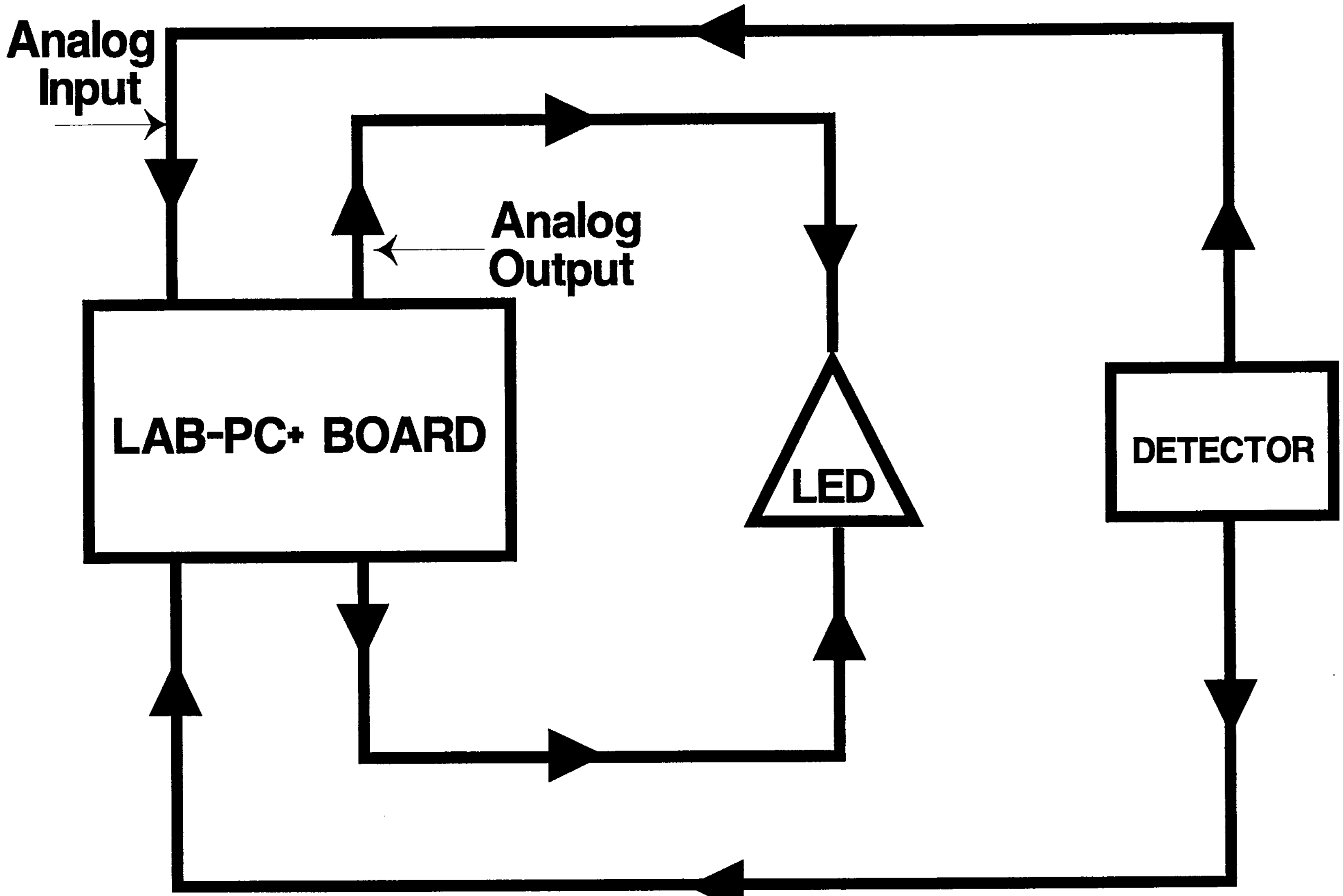


fig. 1

THE ONE THAT IS GOING TO WORK.VI

(analog input)

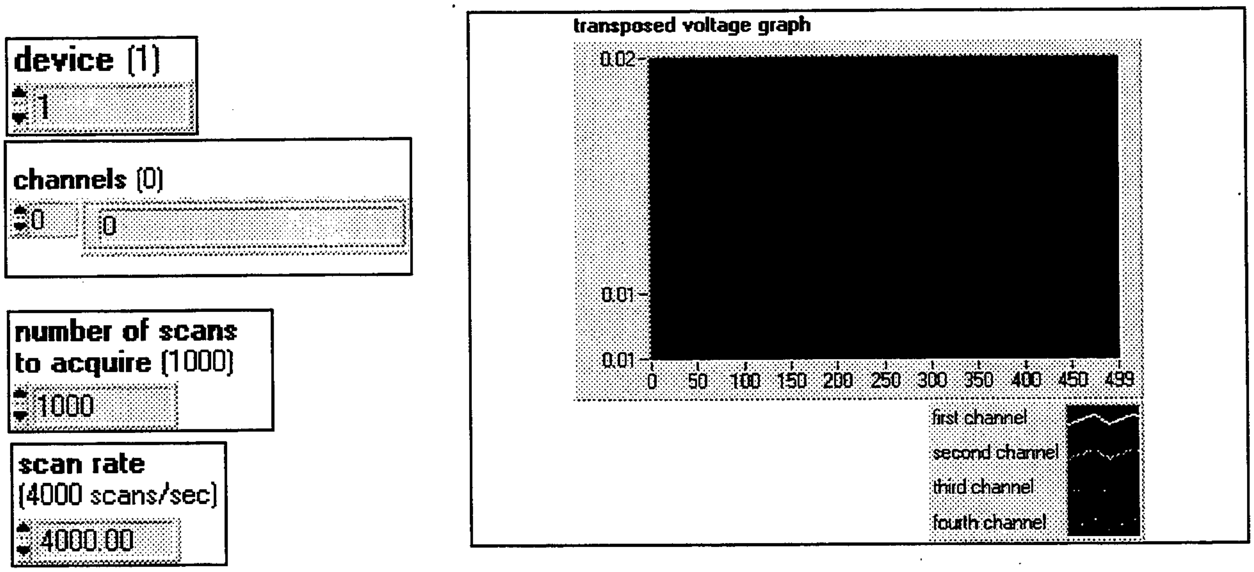


fig. 2

FUNCTION GENERATOR.VI

(analog output)

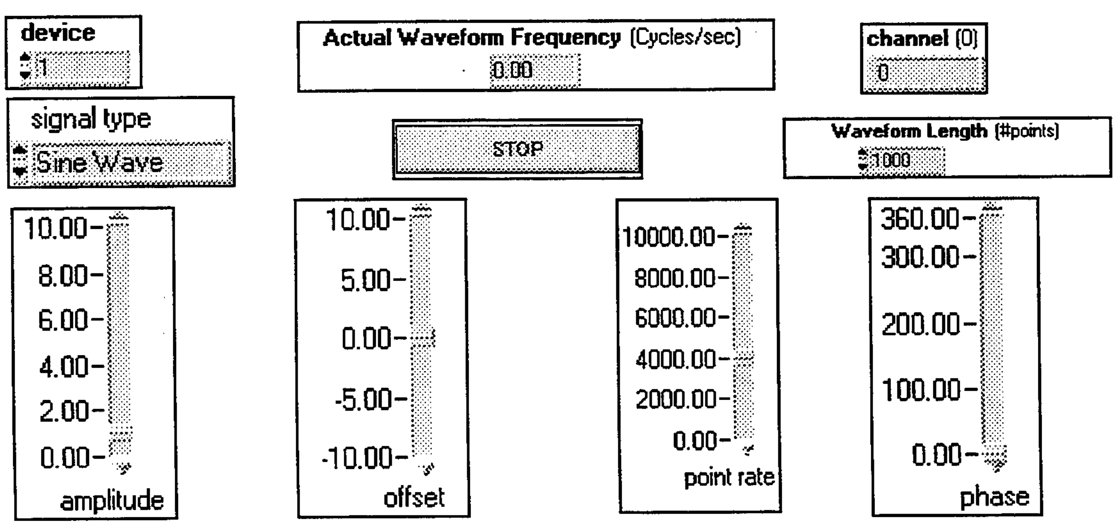


fig. 3

SUBVI COMPUTE WAVEFORM.VI

(analog output)

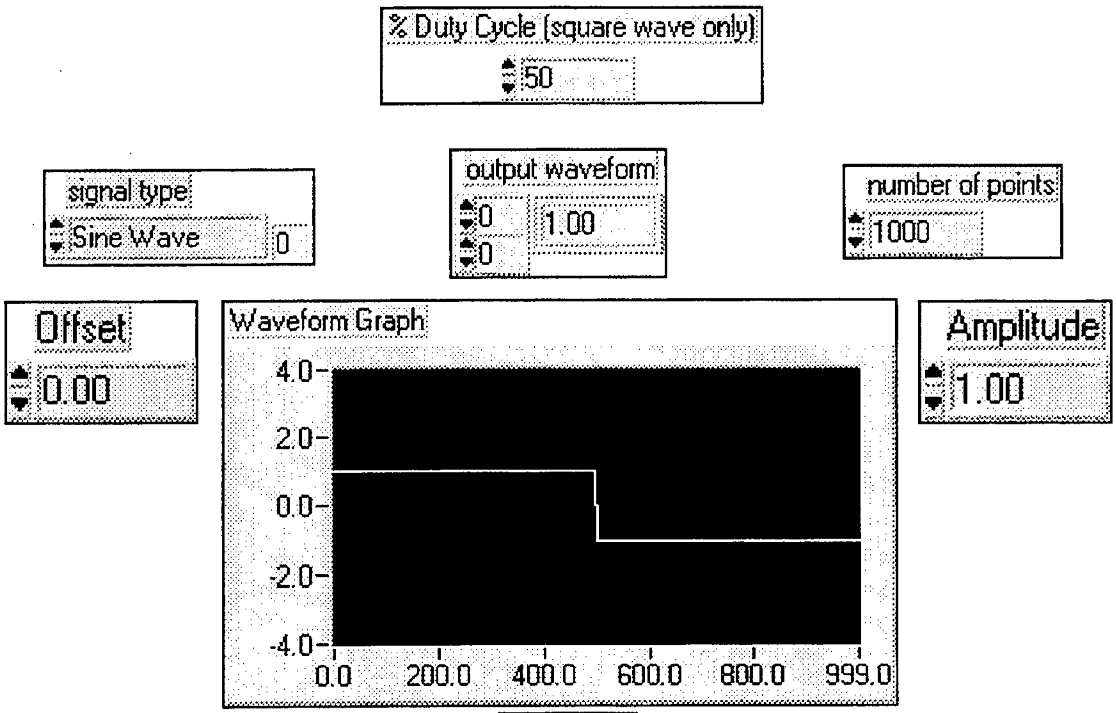


fig. 4

THE FINAL EPISODE OF DAQ.VI

(analog output/input)

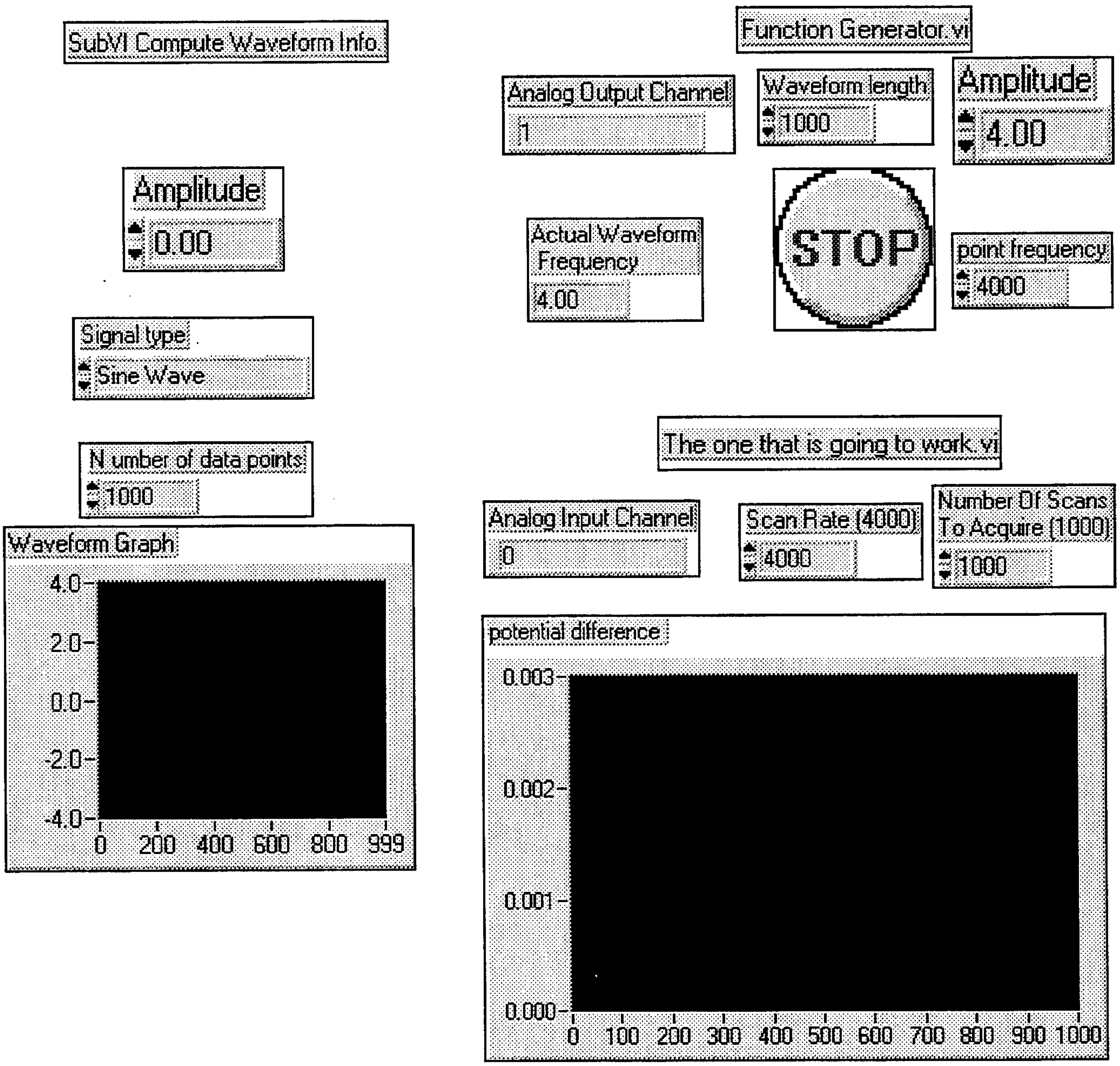


fig. 5

The final episode of DAQ.vi
08/09/95 07:51 AM

Front Panel

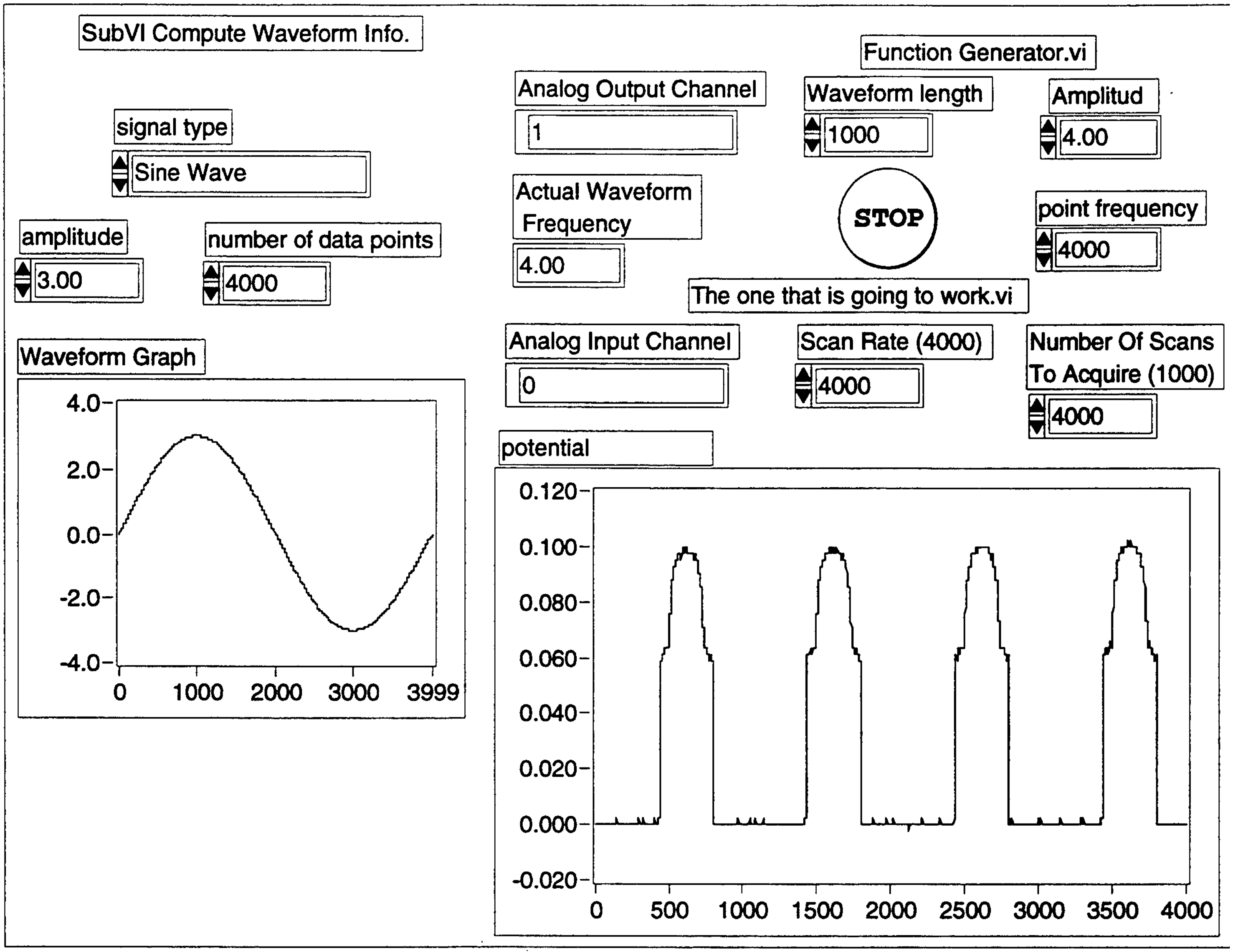


fig.6

SPEECH PROCESSING

Patrick R. Stout

Camden Central High School
Oswego Street
Camden, NY 13316

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

SPEECH PROCESSING

Patrick R. Stout
Camden Central High School

Abstract

During my tour at Rome Laboratory, I was involved in three different projects. For each project, I supported the engineers and developers in the Speech Processing Laboratory. The first was the creation of a database for audio recordings that were obtained on a field data collection. The second was the preparation of audio data from a 911 call. The third project was the creation and inspection of audio data for a verification system.

SPEECH PROCESSING

Patrick R. Stout

Introduction

In recent years, speech processing has taken steps toward developing and applying new speech technologies. Among these technologies are speaker verification and speaker identification. The members of the Speech Processing Laboratory use these technologies for many of their projects. These projects consist of identifying a person on the basis of his or her own voice print, field data collection, and verifying a speaker on the basis of their voice print. I assisted the engineers by creating a database for the field data collection. Also audio data from a 911 call needed to be marked up and recorded onto a Digital Audio Tape (DAT). This audio data was used to determine if two speakers were the same person. A third project I worked on was the in-house collection and inspection of audio data to be used with a voice verification system.

Methodology

In trying to accomplish these tasks many different kinds of software and hardware. The software included Microsoft Access, Turtle Beach's Waves, Microsoft Excel on a Macintosh, and X-Waves on a UNIX work station. The hardware used included a DAT Recorder, Macintosh, Personal Computer, and Sun Sparc Stations connected to a local network. With the availability of these software and hardware systems the tasks were easily overcome. First was the creation of a database. While members of the Speech Lab were collecting audio data someone needed to create a database. Never creating a database before, I learned about the different options and functions that are included in making a database. The purpose of this task was to aid the engineers in the creation of a database for their audio recordings and other information. In my attempt to create a database I found guidance from the on-line interactive help to be very useful using Microsoft Access. Microsoft Access uses helpful options such as the cue cards and a help menu. After reading through the help options I was able to create a variety of forms and reports. These forms and report included the equipment used, the platforms recorded, and the audio samples recorded along with their visual appearance. A query was also created that sorted out the platforms

recorded. When the engineers came back to input their data into the database there was something they could work with. I gained much knowledge about the structure and purpose of a database.

People often call the 911 service to ask for help or give it. In this case a woman called up 911 and told them that the suspect the police were holding was the right person. The woman however didn't leave her name or any other information about herself. The police think they know who the caller is and they want the Speech Lab to use speaker identification between the 911 call and a volunteer who they think is the caller. They brought in a volunteer who they suspect made the 911 call. The volunteer was asked to repeat the words and phrases that were on the 911 tape. The original DAT tape, with the actual 911 recording, was transferred into a .wav file using Turtle Beach's WAVE software. The next action taken was marking the audio data. The tape containing the volunteer's audio was also transferred into a .wav file using Turtle Beach's WAVE software and it was also marked like the 911 suspect audio data. Once both files were marked they were recorded back onto the DAT tape with three seconds between each word and ten seconds between each person. Then each word from each speaker was recorded ten times with three seconds in-between each repetition and five seconds in-between each word. The end result was a DAT tape sectioned off in an organized way so it can be used to check and determine if the volunteer and the 911 speakers were one in the same.

In developing new way to use technologies, the Speech Lab has discovered a way to create and use a door verification system. This system will enable only authorized personal into the Speech Lab. The door verification system will consist of a telephone, located outside the door to the lab, to act as the input device so the voice print can be processed, a database of audio files used to check the speakers voice against. Verification software determines if the speaker's voice print matches any others in the database. I was instructed to create smaller three second audio files from a larger audio file containing all data collected. Each repetition of each word from each speaker was saved as an individual three second audio file. This was accomplished by using the X-Waves software running on a UNIX operating system. There were a total of thirty-one speakers who said seven words ten times each. Each speaker was suppose to say each repetition with a couple of seconds in-between, but unfortunately this wasn't the case. Some speakers didn't say words the right number of times and some said the words too fast or slow. This caused the job of cutting the audio up into three second files more difficult. Some individual files turned out to be

as small as two seconds or smaller. Once the data had been cut up and saved, it was then time for it to be Q.C.'d or checked to see if it was acceptable. For this I used Microsoft Excel on a Macintosh to create a spreadsheet checklist. The audio files were then looked at one by one until they were checked and O.K.'d by different people. After that the files were copied onto an optical disk. While on the optical disk, they were transferred into raw binary files with their headers removed. The results ended with 2,170 raw binary files ready to be further processed.

Conclusion

My stay in the Speech Processing Laboratory was an enriching one. With their dedication the accomplishments were more abundant than could be expected. The technologies they are developing are going to be used in civilian as well as military applications. Speaker verification has many potential roads it can follow, such as secure entry or restricting usage of products by authorized personal only. Speaker identification is a valuable application. With it, the proper authorities can get that beneficial edge they need. Finally, with constructed databases test data can be found easier for what ever purpose it can be used for. Although no final product was produced, through my help steps toward that final produce will be easier and more quickly accomplished. I completed my tasks on the team projects I was involved in. Data is now ready to be used for experimentation by engineers in the Speech Processing Laboratory.

Electrical and Magneto-optical Measurements of Doped and Undoped InP Thin Films

Lauren M. Theodore

Lincoln-Sudbury Regional High School
390 Lincoln Rd.
Sudbury, MA 01776

Final Report for:
High School Apprentice Program
Rome Laboratory

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

August 1995

Electrical and Magneto-optical Measurements of Doped and Undoped InP Thin Films

Lauren M. Theodore
Lincoln-Sudbury Regional High School

Abstract

Electrical and magneto-optical measurements were made of transition metal- and rare earth-doped indium phosphide (InP) thin films grown on (100) InP substrates. Dopants included manganese, terbium, erbium and europium. Electrical properties of the films (the carrier concentrations, mobilities and resistivities) were measured using the Hall and van der Pauw techniques. The magneto-optical property of interest was the Faraday effect. In general, the rare earth- doped films were n-type with carrier concentrations on the order of 10^{17} cm^{-3} and mobilities around 3000 cm^2/Vs. As expected, at 77K the carrier concentrations of the films decreased while the mobilities increased. Doping with manganese caused the films to become p-type. Both erbium and europium additions to the melt caused a decrease in carrier concentrations by an order of magnitude due to gettering effects. However, erbium appears to be a much more effective gettering agent than europium because, after the initial addition of rare earth to the melt, subsequent erbium-doped films continued to maintain low carrier concentrations, whereas europium-doped films did not. Also, the mobility increased by several orders of magnitude after erbium was added to the melt. In studying the Faraday effect, both undoped and iron-doped InP samples were measured to have Verdet constants of 4.4°/T/mm and 5.9°/T/mm, respectively. As expected, a linear relationship was shown to exist between the degree of rotation and the path length. By subtracting the measured substrate rotation from the measured rotation of a doped film, a Verdet constant of approximately 22.7°/T/mm was calculated. As shown by these measurements our technique appears to be an effective method for measuring Faraday rotation.

Electrical and Magneto-optical Measurement of Doped and Undoped InP Thin Films

Lauren M. Theodore

Introduction

Transition metal and rare earth doped $In_{1-x}Ga_xAs_{1-y}P_y$ thin films were studied for use in magneto-optic (MO) applications, such as Faraday rotators. Both the Hall and van der Pauw techniques, along with analysis of the Faraday effect, were used to characterize the properties of these epitaxial thin films. The Faraday effect could have many uses in opto-electronic integrated circuits (OEICs) which may be used for optical communications[1-3], optical signal processing[4] and optical data storage[5]. Currently, magneto-optical materials, such as yttrium iron garnet (YIG)[6], are used in isolators[3], modulators[1] and waveguides[2]. However, growing YIG films requires high temperatures and garnet substrates, which are incompatible with indium phosphide and other III-V semiconductors that melt at much lower temperatures[2]. InP substrates would enable lithographic techniques to be used in patterning devices and integrating MO devices with other III-V devices, such as lasers, waveguides, and detectors. The object of this research, therefore, is to find magneto-optical films that are compatible with an indium phosphide substrates.

Electrical Properties. In order to measure the carrier concentrations, mobilities and resistivities of samples, the Hall and van der Pauw techniques are used. For the Hall measurements, 4 probes are put down on the corners of a square, uniform sample. If the film is not uniform, the readings will not be accurate. A current is applied diagonally between two probes across a sample which is placed in an electromagnet, Figure 1. Current is defined as the movement of positive charge, and therefore, the positively charged holes move in the direction of the current, I, and the electrons move in the opposite direction.

With the magnetic field, B, applied orthogonally to the sample, the moving charge is subjected to a Lorentz force, F_L, causing the charge to curve in the following way:

$$\mathbf{F}_L = e\mathbf{E} + e\mathbf{v}\times\mathbf{B} \tag{1}$$

where e is the charge on an electron; **E** is the electric field vector; **v** is the electron velocity vector and **B** is the magnetic field strength vector. Thus there is a build-up of charge and the resulting electric field can be used to

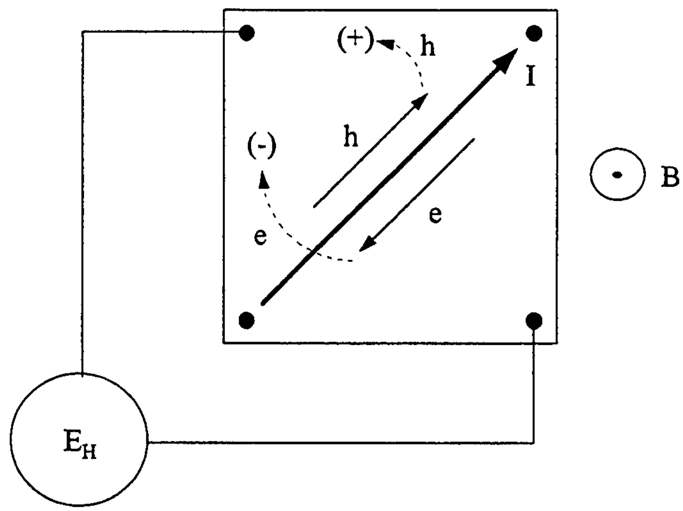


Figure 1- Measurement of the Hall effect.

calculate carrier sign and concentration. This field (E_H), known as the Hall voltage, is perpendicular to both the current and the magnetic field. A Hall coefficient, R_H, can be defined

$$R_H = (E_H / JB) = [E_H / (qnvB)] \qquad (2)$$

where J is the current density, B is the magnetic field, n is the carrier concentration, q is the carrier charge, and v is the drift mobility. A derivation of equation 2 from equation 1 can be found in the literature[7].

The carrier concentration and sign can be calculated when the Hall coefficient is known. In order to measure mobility, the resistivity must be known. Resistivity is measured using the van der Pauw technique wherein a current is applied to two probes on one side of the square sample, creating a voltage which is read at the opposite two probes. The eight voltages taken are used to calculate two resistivity values, p_A and p_B, which are then averaged to get one final resistivity. Once the resistivity is known, the mobility can be determined as follows. The conductivity of the sample can be found using the equation

$$\rho = 1 / \sigma \qquad (3)$$

where ρ is resistivity and σ is conductivity. Using the following equation for conductivity, the mobility, μ, can be calculated.

$$\sigma = [ne\mu_e + pe\mu_h] \qquad (4)$$

where e is the charge on an electron, n and p are the concentrations of electrons and holes.

Magneto-optical Properties. A magneto-optical set-up was designed to measure the rotation of light caused by the Faraday effect. This effect was first discovered in 1845 when Michael Faraday noticed a rotation when plane-polarized light was sent through a block of glass parallel to a magnetic field. The effect occurs because of an inequality between the refractive indices of left- and right- circularly polarized light. Plane-polarized light can be expressed as two equal, but opposite circular components- that is, a left- and a right-circular polarization. The polarization which is traveling with the magnetic field will travel slightly faster than the polarization moving against the field. This effectively rotates the plane-polarized light in the direction of the magnetic field.

The amount of rotation is proportional to the strength of the magnetic field, the length of the light's path, and to the Verdet constant, V-the material property used to compare Faraday rotators- as shown:

$$\theta(\lambda) = V(\lambda)\, \ell\, \mathbf{B} \tag{5}$$

where θ is the angle of rotation, λ is the wavelength of the light, ℓ is the distance of the light's path, and $\mathbf{B}$ is the applied magnetic field. One appeal of the Faraday rotation is that it is a non-reciprocal property. For example, light which passes through an isolator (composed of a Faraday rotator and polarizer) located in front of a laser will be rotated. But, any of this rotated light reflected back to the laser through the isolator will continue to be rotated further from the initial polarization. The reflections will be blocked by the polarizer and will not be able to damage the laser.

Experimental

Electrical Properties. The electrical properties of thin films grown on indium phosphide substrates were characterized by the Hall and van der Pauw techniques. To begin sample preparation, the substrate/film samples were cleaved into 1 cm^2 squares using a SYS Manual Scriber manufactured by Kulicke and Soffa. Small pieces of indium metal were then cut and positioned as contact points in the four corners of each sample using teflon tweezers. To encourage diffusion of the indium metal into the sample, each sample was annealed with a rapid thermal anneal (RTA) for 30 seconds at 350°C. The cut and annealed sample was placed in a tray, Figure 2. Four Alessi mobile microprobes, connected to a Keithley 705 Scanner, were positioned around the tray.

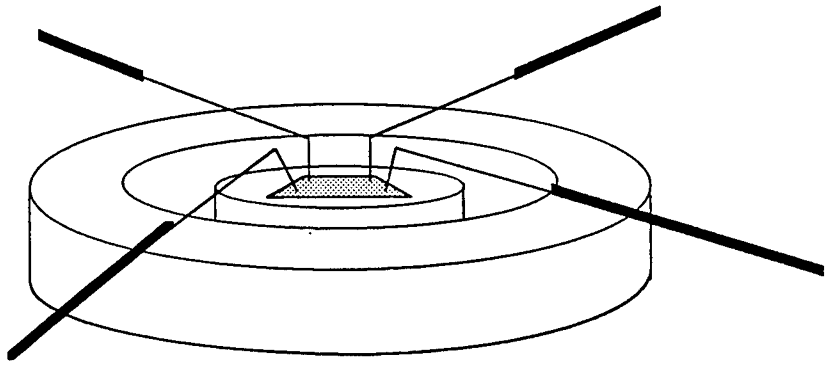


Figure 2- Probe placement for Hall/van der Pauw measurements.

Each probe was then aligned over one contact point. Aligning the probes was a difficult task because the contacts were only about $1mm^2$ and the probe tips themselves were very fine. When lowering the probes onto the contacts, it was important that the surface of the sample was not scratched, because the film was only about a micron thick and could be easily damaged. Probes diagonal to one another were lowered consecutively to ensure that pressure put on the sample was balanced.

For the Hall measurements, the tray was slid into a Walker Scientific electromagnet. The magnetic field was set to 3380 gauss in order to comply with the computer program already in existence. The Hall/van der Pauw program was then run on a computer which controlled all of the instruments except the magnet and which completed all measurements and calculations. The program needed a specified resistivity setting, current, and number of readings to be taken for each measurement. The resistivity was set to "high". The current chosen had to be between 0.5 pA (picoamperes) and 2 mA. It was best to use as large a current as possible, in order to maximize the accuracy of the results. For most of the samples in this project the maximum current was found to be 1×10^{-5} A. Two readings were taken for each data point.

Four measurements were taken in the +B magnetic field before the computer prompted for the field to be switched. On the Walker Scientific magnet there was a reverse button which would automatically switch the field from +B to -B. Four measurements were then taken in the new field. For van der Pauw measurements, the sample had to be taken out of the magnetic field. At first, the sample was left in the magnet and the magnetic field was ramped down to 0 gauss. A more convenient method was to slide the sample out of the magnet to take the van der Pauw measurements and hit the reverse button so that the magnet was switched to the +B field for the next Hall

measurements. There seemed to be no difference between these two methods. The flux at the sample was measured to be zero in the second method with a gaussmeter. The computer print-out of the results gave the averages taken for each of the eight Hall voltage measurements and the eight van der Pauw measurements, the Hall and van der Pauw coefficients, the average resistivity, the carrier concentration and the mobility.

The Hall/van der Pauw measurements were taken for each sample, not only at room temperature, but also at 77K. Before the sample was slid into the magnetic field, liquid nitrogen was poured into the teflon tray where the sample sat. Occasionally, at this temperature, the current had to be lowered to 1×10^{-9} A in order to obtain a reading. After every run with the liquid nitrogen, the sample and tray were blown off with an air gun and warmed with a heat gun if needed.

Magneto-optical Properties. The magneto-optical set-up used to measure Faraday rotation is shown in Figure 3. An infrared laser at 1.064 μm was used to pass light through an analyzer or rotator, a chopper, the sample which was mounted in an electromagnet and a beam splitter. The beam splitter divided the light into TE and TM components which were detected by two InGaAs detectors. The detectors and chopper, which cut noise from the signal, were connected to a lock-in amplifier whose output was sent to a strip chart recorder. Either the analyzer or rotator was used to rotate the light, however, in the initial measurements, the analyzer was found to be more accurate. The paper in the strip chart recorder ran at 3 cm/sec. and the input on the recorder was set at 2 V.

The light was oriented such that the TE polarization was 45° from vertical and the analyzer was set to allow the maximum signal to pass. In this configuration, equal amounts of TE and TM polarizations were passed through the sample and the detectors read equal signals. The lock-in amplifier was set to read the difference which was approximately zero. Small adjustments made with the analyzer "zeroed" the signal.

There was one caution taken with zeroing the signal on the lock-in. The functions of the TE and TM modes were periodic and intersected at their minima and in between their maxima, Figure 4. The lock-in will read zero at any of these intersections. In order to get an exact reading of the rotation, it was necessary to study the intersections between their maxima. At these intersections, the slopes of the functions were very steep and, therefore, any slight change in the angle of the analyzer will resulted in a drastic change in intensity.

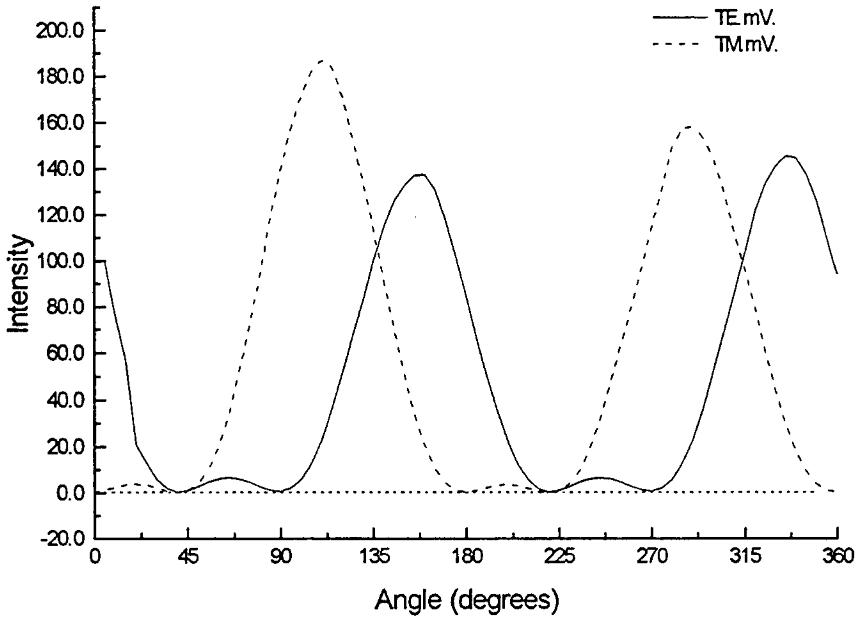


Figure 4- Intensity of the TE and TM modes versus the angle of the light.

The sample was aligned in the electro-magnet until the signal from the laser was maximized. The lock-in amplifier was set to read only the input from one of the detectors. It did not matter whether the lock-in was reading from the TE or TM modes because it was only important to maximize the light passing through the sample. After the sample was aligned, the signal was then zeroed. The lock-in was set to read A - B, or TE-TM, and by adjusting the analyzer, the angle at which equal amounts of TE and TM modes passed through the sample was found.

Since at this angle the lock-in read zero cumulative volts, when the magnet was switched on and off, the effect of rotation on the intensity could be amplified. When the lock-in amplifier signal was zero, the recorder pen sat on the center line on the strip chart paper. The strip-chart recorder was calibrated so that, when the light was rotated $\pm 1°$, the signal was at maximum scale. This was done three times to ensure that an accurate average of the range could be taken. The analyzer was then turned back to the initial angle and the strip chart recorder was left to run at this angle- the center line of the graph paper- for about 30 seconds in order to get a stable "zero" line. The electro-magnet was then turned on and off three times with the magnetic field going in the direction of the light.

The direction of the field was then switched so that the field went against the light and the magnet was again turned on and off three times, Figure 5. The Faraday rotation was turned on was found by dividing the average length of the magnetic response on the left by the average length of the 1° rotation to the left; the same was done for the right. The experimental error was calculated.

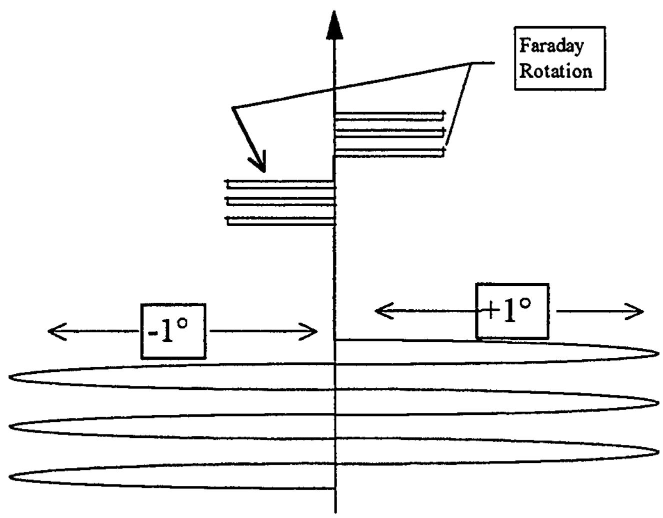


Figure 4- The Faraday rotational effect as shown on strip-chart recorder paper.

Results and Discussion

Electrical Properties. The results of the Hall/van der Pauw measurements are tabulated in Table I. Films from melt #10 are not included in the table because their Hall/van der Pauw results were inconsistent, indicating that the samples were not uniform. In preparing most of the doped films, 0.04 weight % dopant (Eu, Mn, Tb or Er) was added to the melt. Except for the manganese doped films, which were p-type, the films were n-type with carrier concentrations on the order of $10^{17}cm^{-3}$ and mobilities around 3000 cm^2/Vs. At 77K, the carrier concentrations were slightly lower and the mobilities were slightly higher than those at room temperature. This change occurred because, at low temperatures, the thermal energy needed to activate carriers was not available. Whenever the carrier concentration drops, mobility rises because carrier scattering is reduced.

Table I- Electrical properties of thin films on InP						
Sample	**Melt #**	**n or p**	**Room Temp.**		**77K**	
			n/p (10^{17} cm^{-3})	μ (cm^2/Vs)	n/p (10^{17} cm^{-3})	μ (cm^2/Vs)
InP	5	n	1.5	2800	1.3	3453
Eu: InP*	5	n	0.4	3200	0.35	5200
Eu: InP	5	n	1.6	2190	1.32	2860
InP	6	n	3.1	3000	2.1	3231
Mn: InP	6	p	0.216	153	Semi-insulating	
Mn: InP	13	p	0.25	180	Semi-insulating	
InP	12	n	6	630	5.1	614
Er: InP	12	n	0.004	2780	0.0041	25600

* First film after addition of Eu to the melt.

In melt #5, the added Eu caused the carrier concentration to go down by almost an order of magnitude. Eu, along with most rare earths, has a high chemical affinity and is an excellent gettering agent. It will, therefore, form compounds with donor impurities in the melt such as carbon and silicon. These compounds usually form a slag and are no longer part of the melt. It appears, however, that there was a source of impurities, possibly in the boat and/or cylinder where the films were grown, because the carrier concentration returned to its original value after the initial addition of Eu.

In melt #6, manganese was added, causing the films to become p-type. Because the preferred valence state of manganese is Mn^{2+}, the manganese will become an acceptor when in an In^{3+} site in order to neutralize the overall charge. These Mn-doped films became semi-insulating at 77K because manganese is a deep level impurity. In other words, its energy state is in the middle of the band gap so a large amount of energy is needed for thermal activation. The number of holes, p, is a function of temperature, T, as follows

$$p \propto \exp(-E_a / kT) \tag{6}$$

where E_a is the activation energy; k is the Boltzmann constant. As the temperature is decreased, for example to 77K, the carrier concentration also decreases because, from equation 3 and equation 4, resistivity (ρ) is inversely proportional to carrier concentration (p) as shown in the following equation

$$1 / \rho = pe\mu_h \tag{7}$$

where e is the charge on an electron and μ_h is the mobility of the holes. Note that the films are p-type, so $ne\mu_e$ from equation 4 is negligible compared to $pe\mu_h$. Therefore, as the carrier concentration decreases, the resistivity increases. In melt #13, 0.1 weight % of manganese was added to the melt but only slightly more manganese appeared in these films than in melt #6 films. Therefore it appears that manganese saturates the melt at around 0.05 weight %. Melt #13 films also became semi-insulating at 77K for reasons previously stated.

The undoped InP films from melt #12 had very low mobility at both room temperature. It is likely that p-type impurities are present in the film which charge compensate the n-type impurities. This causes increased impurity scattering and therefore decreased mobility. Erbium additions to the melt caused a decrease in the carrier concentration as did europium. However, Er appears to be a more effective gettering agent than Eu because the carrier concentration stayed low in several subsequent films of melt #12. Also, the severe rise in mobility in the Er-doped films indicates that they have been nearly purified.

Magneto-optical properties. An initial test of the magneto-optical measurement technique involved looking for differences between two bulk samples with uniform thicknesses, undoped and iron-doped InP. There was, in fact, a difference between the measured Faraday rotations of the two. As expected, the iron-doped InP had a larger Faraday rotation than the undoped InP. For the undoped InP a rotation of 4.4°/T/mm was measured and for the iron-doped InP a rotation of 5.9°/T/mm was measured. With other methods, no difference between these two samples was detected. The current measurement technique is therefore a comparatively effective method of measuring Faraday rotation.

The effect of thickness on the degree of rotation was observed. According to equation 5, there should be a linear relationship between the two. Three samples of different thicknesses from the same iron-doped InP wafer were tested. Each of the samples had been polished for different lengths of time. The thinner samples transmitted less light, assumingly from scattering caused by surface damage. In order to increase the gain from the lock-in amplifier, the full-scale was decreased, thereby increasing the sensitivity. If a linear relationship existed between θ and ℓ, K, the proportionality constant, should have been the same for all three samples. Following equation 5, K

is equal to VB, the Verdet constant multiplied by the strength of the magnetic field, and these two factors were the same for all three samples. However, varying the lock-in amplifier sensitivity was found to cause the proportionality constant to vary. When Group 1 and Group 2 were measured again at the same lock-in sensitivity (full scale= 20mV), K was calculated to be 0.658 and 0.642, respectively. This corresponds to a Verdet constant of approximately 5.9°/T/mm.

Table II - Effect of lock-in sensitivity on measurements

Sample	LI* full scale	length (ℓ)	θ	K
Group 1	20 mV	0.45mm	0.368	0.818
Group 2	5mV	0.38mm	0.234	0.616
Group 3	20μV	0.27mm	0.153	0.567

* LI- abbr. for lock-in amplifier

As in the samples above, a dependence of the angle of rotation on the lock-in sensitivity was seen in sample M10-6 and its substrate, Wafer 11. However, several consecutive measurements of M10-6 and Wafer 11 at the same lock-in sensitivity (full scale= 20mV) showed that M10-6 demonstrated a greater rotational effect than Wafer 11. For Wafer 11, θ was $0.240 \pm 0.005°$ and for M10-6, θ was $0.265 \pm 0.004°$. According to equation 5, an increase in rotation could be due to the Verdet constant (V), the magnetic field (B) or the path length (ℓ). The strength of the magnetic field is a constant, but the Verdet constant was expected to increase because of dopants, Tb and possibly Mn, in the M10-6 film, and the path length was longer for M10-6. In order to show that the increased rotation was not due simply to the change in path length, the greatest possible rotation caused by a change in thickness was calculated. The substrate for M10-6 was 350μm (equal to the thickness of Wafer 11), and the maximum possible thickness of the M10-6 film was 10μm. The Faraday rotation of Wafer 11 was measured to be 0.240°. If V and H stay the same (ie. the sample is of identical make-up, and the field strength remains the same) then a change in the sample thickness from 350μm to 360μm would result in a rotation of 0.247° yielding a difference of 0.007°. However, the rotation of M10-6 was measured to be 0.265°, yielding a difference three times greater than the maximum difference that could be expected due to increased thickness alone. The field had not changed, and the change in path length only accounted for a fraction of the rotation measured, so the Verdet

constant has to have been increased by the presence of dopants. This not only further reaffirms that the technique is effective, but also shows that the dopants added, terbium and possibly manganese, do change the magneto-optical properties of the films. In fact, by subtracting the measured substrate rotation from the measured rotation of the M10-6 film, the Verdet constant of the film can be calculated to be approximately 22.7°/T/mm.

Conclusion

The transition metal- and rare earth- doped thin films characterized in this research show promise as future magneto-optical devices. In order to measure the electrical properties of the films, the Hall and van der Pauw techniques were used. The Faraday effect was the magneto-optical property studied. Rare earth- doped films were n-type and had carrier concentrations on the order of $10^{17}cm^{-3}$ and mobilities of around 3000 cm^2/Vs. At 77K, the carrier concentrations of these films decreased and, therefore, the mobility increased. Manganese-doped films were p-type at room temperature, but became semi-insulating at 77K. Erbium was found to be a much more effective gettering agent than europium because erbium continued, through consecutive films, to keep carrier concentrations low, and carrier concentrations in europium-doped films increased in films after the first. Not only does the order of magnitude drop in carrier concentration show that erbium is an excellent getterer, but the mobility in the erbium-doped sample increased from 2780 cm^2/Vs to 25,600 cm^2/Vs.

Initial magneto-optical measurements of the Faraday effect were taken using undoped and iron-doped InP. The undoped InP had a rotation of 4.4°/T/mm whereas the iron-doped InP had a rotation of 5.9°/T/mm. A linear relationship was found to exist between the degree of rotation and the path length. The Faraday rotation of a doped InP film was measured to be approximately 22.7°/T/mm. These measurements indicates that our measurement technique is effective.

References

[1] K. Matsuda, H. Minemoto, O. Kamada, S. Ishizuka, *Appl. Opt.* **27** 329 (1989).
[2] K. Matsuda and S. Ishizuka, *Appl. Opt.* **30** 1963 (1991).
[3] M. Shirsaki, H. Nakajima, T. Obokata, K. Asama, *Appl. Opt.* **23** 4229 (1982).
[4] S. Donati, V. Annovazzi-Lodi, T. Tambosso, *IEEE Proc.* **135** [J5] 372 (1988).
[5] T. Suzuki, C. Lin, A. Bell, *IEEE Trans. Magnet.* **24** 2452 (1988).
[6] D. Gualtieri, *J. Appl. Phys.* **67** 4793 (1990).
[7] L.L.Hench and J.K.West, Principles of Electronic Ceramics, (John Wiley & Sons Inc, New York, 1990) p. 81.